Robustness To Visual Perturbations In Pixel-Based Tasks

A Dissertation
Presented to
The Academic Faculty

by

Dylan Dexiong Yung

In Partial Fulfillment
of the requirements for the Degree
Master of Science in Computer Science
College of Computing

Georgia Institute of Technology

May 2023

Robustness To Visual Perturbations In Pixel-Based Tasks

Approved by:

Dr. Zsolt Kira, Advisor
College of Interactive Computing
Georgia Institute of Technology

Dr. Judy Hoffman, Co-Advisor
College of Interactive Computing
Georgia Institute of Technology

Dr. Danfei Xu
College of Interactivate Computing
Georgia Institute of Technology

Date Approved: January 13, 2023

Acknowledgements

I would like to acknowledge my labmates **Junjiao Tian**, **Andrew Szot** and **Prithvijit Chattopadhyay** for their contributions and efforts to help me improve as a researcher. A special thanks to **prof. Zsolt Kira** for being a wonderful mentor and spending so much time pushing the growth of his students. Other thanks to **prof. Judy Hoffman** for her insights and expertise in helping make these works come to fruition.

Table of Contents

List of Tables

List of Figures

Summary

Convolutional Neural Networks (CNNs) have been shown to provide great utility across many vision tasks and have become the go-to model for problems involving video or image input. Though they've shown promise across many problems they come with inherent flaws. For example, in image classification, CNNs are known to output very high confidence values even when their accuracy is low. This is exacerbated when visual perturbations are introduced to inputs causing accuracy to drop, but confidence to remain high. This is similarly problematic when models use visual inputs for decision-making, such as through pixel-based Reinforcement Learning (RL) where an agent must learn a policy leveraging images of the environment as input. RL agents under these settings can perform well in training, but once deployed may face unseen visual perturbation, causing an erroneous execution of their learned task. Poor robustness to the previously mentioned examples is deadly in applied Machine Learning (ML) in the medical field and autonomous vehicles. Thus ways to impart robustness on CNNs for image classification and RL are of utmost importance. In this thesis, we explore solutions to the problem of *overconfident image classification models* and *embedding robustness to visual perturbations in RL*. We propose two distinct frameworks for doing so in two contexts: Image-based classification (**Geometric Sensitivity Decomposition (GSD)**) and decision-making (**Augmentation Curriculum Learning (AugCL)**).

CNNs utilized for image classification has been shown to be erroneously over-confident. A large contributor to the overconfidence is attributed to a combination of Cross-Entropy loss, the standard loss for classification, and the final linear layer typically in vision models. GSD decomposes the norm of a sample feature embedding and the angular similarity to a target classifier into an *instance-dependent* and an *instance-independent* component. The instance-dependent component captures the sensitive information about changes in the input while the instance-independent component represents the insensitive information serving solely to minimize the loss on the training

dataset. Inspired by the decomposition, we analytically derive a simple extension to current softmax-linear models, which learns to disentangle the two components during training. On several common vision models, the disentangled model outperforms other calibration methods on standard calibration metrics in the face of out-of-distribution (OOD) data and corruption with significantly less complexity. Specifically, we surpass the current state of the art by 30.8% relative improvement on corrupted CIFAR100 in Expected Calibration Error.

Pixel-based RL has shown a lack of ability to identify and learn visual features when things such as color have been changed. Image augmentation has been shown to add to this, but is difficult to balance. AugCL is a novel curriculum learning approach that schedules image augmentation into training into a weak augmentation phase and a strong augmentation phase. We also introduce a novel visual augmentation strategy that proves to aid in the benchmarks we evaluate on. Our method achieves state-of-the-art performance on Deep Mind Control Generalization Benchmark when combined with previous methods. Code available at
https://github.com/GT-RIPL/Geometric-Sensitivity-Decomposition.git
and
https://github.com/GT-RIPL/AugCL.git.

Chapter 1 Introduction

The importance of calibration in vision models is in situations where incorrect inference can be costly such as in the medical field, users of vision models need an accurate sense of how confident the model is in its prediction. On the other RL is typically deployed in the real world which involves distribution shifts such as lighting and color changes, which the agent must also be robust to as an erroneous execution of a policy can be dangerous as well. Calibration and robustness are not inherently built into vision models and thus require auxiliary methods which this thesis focuses on.

Image classification models are trained and validated on data from the same distribution. However, in the real world sensors degrade and weather conditions change. Similarly, subtle changes in image acquisition and processing can also lead to a distribution shift of the input data, and will typically decrease the performance (e.g. classification accuracy). However, it has been empirically found that the model's **confidence** or probability of prediction remains high even when accuracy has degraded (29). The process of aligning confidence to accuracy is called **Calibration**. Calibrated probability provides valuable uncertainty information for decision-making. For example, knowing when a decision cannot be trusted and more data is needed is important for safety and efficiency in real-world applications such as self-driving (10) and active learning (67).

Indeed, **Reinforcement Learning** (RL), which relies on raw observations (e.g. images) to perform decision-making, has been shown to degrade in performance under distribution shift as well. An example of distribution shift in RL would be autonomous vehicles where an agent is expected to safely navigate from one point to another and can do so under clear weather, which it was trained under, but unable to under cloud weather (the distribution shift). Several existing approaches to training more robust agents include **Domain Randomization** (54; 64) and **Data Augmentation** (26; 25; 17). Domain randomization modifies the training environment simulator to create more varied training data, whereas data augmentation deals with augmenting the image observations

representing states without modifying the training simulator itself.

We aim to tackle the confidence calibration problem by taking a geometric approach. The final linear layer of vision models is disentangled into two components and the main culprit for over-confidence is calibrated using Temperature Scaling (22). We call our method **Geometric Similarity Decomposition(GSD)**. Our method is backed by recent theoretical intuitions discovered about CNNS (13).

We deal with robustness in RL with pixel-based augmentation. We disentangle augmentations used for improving policy learning and augmentations used for robustness to visual perturbations into two separate networks. We create a curriculum in order to make this disentangled learning possible as networks trained under augmentations can hinder policy learning if not delicately balanced.

We specifically aim to solve these issues in a zero-shot manner, i.e., where distribution-shifted data is unavailable during training. We evaluate GSD's calibration using a corrupted version of the data it's trained on with differing corruptions and severities. AugCL is evaluated on the same task but with simulator modifications that change lighting, color, and background.

Chapter 2 Preliminaries

The main contributions of this thesis are at the intersection of computer vision, RL, and data augmentation. This section is to provide a necessary foundation for those unfamiliar with these topics but assumes the reader has a basic understanding of probability theory, machine learning, and statistics. Preliminaries are broken up into 2 sections: Computer Vision 2.1 and Reinforcement Learning 2.11, which are designed to give the necessary foundations for Ch. 5 and Ch. 6 respectively.

2.1 Computer Vision

For those familiar with Deep Learning and Convolutional Neural Networks this section can be skipped. This section goes over the standard architecture for image classification and pixel-based RL used for works mentioned later in this thesis. We assume a basic understanding of Deep Learning and do not cover optimization for the sake of brevity.

2.2 Images To 3D Arrays

We first discuss the format that images must take in order to be processed by Deep Learning methods. Typically an image is represented by a 3-dimensional array. The depth dimension (channels) represents color saturation, with indexes 0, 1, and 2 representing red, green, and blue respectively. The 2nd and 3rd dimensions of the array represent the width and height of the image respectively. Each value in the array typically ranges from 0 (no saturation of that color) and 255 (full saturation of the color), and an example is shown in Fig. 2.1.

2.3 Feed Forward Network

In this section, we describe the standard Feed Forward Network, which is typically seen at the end of vision models. We can see an example of an Artificial Neural Network

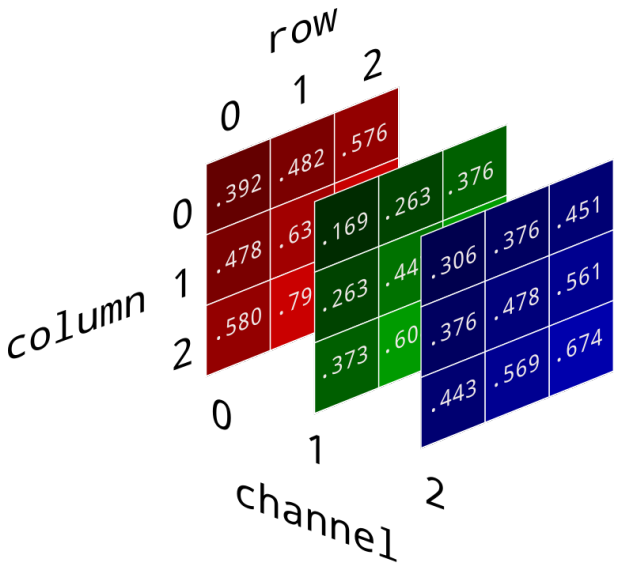


Figure 2.1: Image with 9 pixels, with width and height of 3. Each cell represents color intensity for the respective channel. The ranges have been normalized to [0,1] by dividing the initial values by 255. Image from (58).

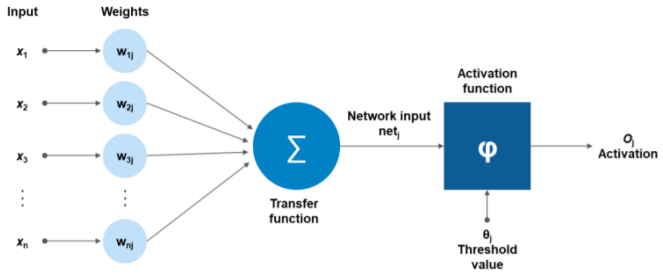


Figure 2.2: Artificial neuron. Image from (20).

in Fig. 2.2. Input data is represented by $x_i, i \in [1, n]$ and **weights** are represented by $w_{i,j}, i \in [1, n]$. j represents the layer index. For simplicity, we'll just assume there is one layer, so $j = 1$. Weights indicate the importance of feature i for the function being minimized. The **Activation Function** is a way to introduce non-linearity and can be chosen from a wide variety of activation functions. Some of them require a threshold value as a hyper-parameter represented by θ_j. An example of an activation function is given in Sec. 2.5. The output of a neuron represents how much "activation" this neuron has achieved. This is inspired by how neurons in the human brain have electricity (activation) running through a pathway of neurons and axons when the brain is recalling information (46). A neuron's activation can be represented as a vector function:

$$f(\vec{x} \cdot \vec{w}, \theta) \tag{2.1}$$

We represent the input as a vector $\vec{x}$ and the weights of the neuron as the vector $\vec{w}$ in eq. 2.1. $f()$ represents the chosen activation function with θ as the threshold parameter. As we can see the dot product of $\vec{x}$ and $\vec{w}$ are taken and passed through the activation function.

2.4 Fully Connected Neural Network

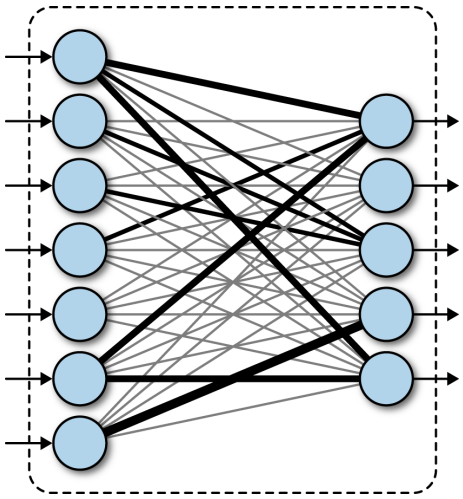


Figure 2.3: Each blue circle represents a neuron and lines signify the passing of activation. Image is taken from (7).

Building upon Sec. 2.3 we define stacking multiple layers. A **Fully Connected Neural Network** is multiple layers of neurons as seen in Fig. 2.3. Each blue circle represents a neuron and we can see that in the second column of neurons, the outputs of the previous layer of neurons are passed as inputs. The primary difference between a single neuron and fully connected multiple layers of neurons is that each layer besides the first applies weight to the output of the previous layer. This allows each neuron to learn the importance of each feature from the previous layer in order to minimize the loss function. Each node in the second layer utilizes Eq. 2.1 but represents all the previous layer outputs as values of $\vec{x}$.

2.5 Rectified Linear Unit

As referenced in Sec. 2.3 we describe an activation function primarily used by all models in this thesis. While there are many activation functions in the field of Machine Learning we'll look specifically at **Rectified Linear Unit (ReLU)** as it is the one primarily used in the works presented in this thesis. As we can see in Fig. 2.4, ReLU forces the activation

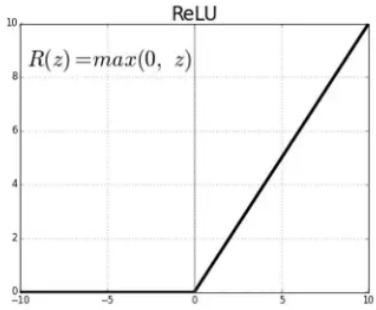


Figure 2.4: ReLU function and graph. Image from (1).

to 0 if the output is less than or equal to 0, otherwise, it'll output the sum of the weights and the input. Due to how optimization of the neural network is performed, as there are increasingly more layers of neurons the gradient can vanish to zero. This is due to calculating the derivative of certain activation functions and how consecutive layers of this cause the gradient to converge to values too close to 0 to calculate. Since ReLU is linear for $z > 0$ it has some nice optimization properties as the gradient is easy to calculate.

2.6 Convolutional Neural Networks (CNN)

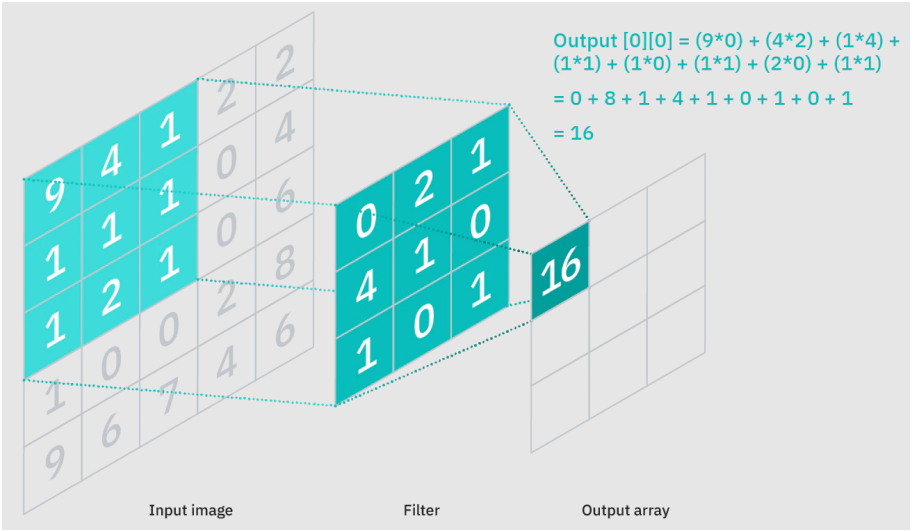


Figure 2.5: Visual example of applying CNN kernel to image. Image from (4)

Now that we've established standard Neural Networks which act as the foundation for Deep Learning, we now define the CNNs the most commonly used models for vision based tasks. While flattening an image into a vector and passing it to a feed-forward network is possible, these networks lack the ability to capture temporal and spatial

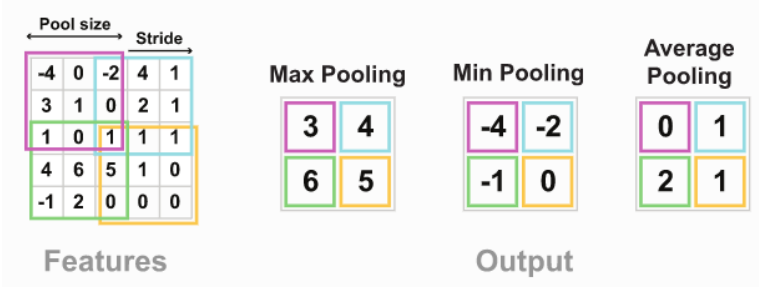


Figure 2.6: Different types of pooling applied to features represented by a 5×5 matrix. All pooling kernels are 3×3. Image is taken from (3).

dependencies. This is where CNNs come into play. CNNs comprise of what is known as a **Filter**, which is typically represented by a multi-dimensional matrix. **Filter Size** determines the width and height of the filter matrix. Each cell of the filter represents a weight similar to a feed-forward network mentioned in Sec. 2.3. An example can be seen in Fig. 2.5. We see that if we have a filter with dimensions 3×3 it is applied to 3×3 sections of the original image. **Stride** determines how many pixels the filter shifts over the input matrix. We can see in Fig. 2.5 that the kernel is applied to a subset of the image of matching size then the values of the kernel are multiplied by their respective cell in the input image. Subsequently, all of these values are summed. Note that the output matrix after applying a kernel is smaller than the original input image. The formula for calculating the output width and height of a CNN kernel applied to an image is:

$$\frac{W - K + 2P}{S} + 1 \tag{2.2}$$

where W represents the width and height of an image (typically in the shape of a square). K represents the kernel size, and P is the padding applied to the image, which is applying added pixels around the outer rim of an image. S represents the stride.

2.7 Pooling Layers

Similar to the Fully Connected Neural Network described in Sec. 2.4, stacking CNNs has been shown to improve performance. An issue with the feature map output of a CNN layer is that they record precise positions of features in the input. So small shifts in the position of features can result in a different output. A common solution to deal with this is **Pooling**. Unlike CNN layers, pooling layers apply a fixed transformation rather than a learned one. Pooling is typically applied to the feature map produced by a CNN

kernel. Examples of different types of pooling and the outputs from applying each can be seen in Fig. 2.6.

2.8 CNN Architecture

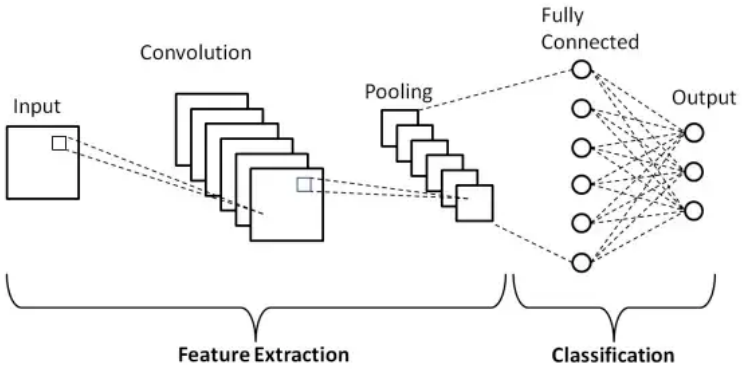


Figure 2.7: Full vision model architecture. Image is taken from (5).

We include Fig. 2.7 which shows a standard vision model architecture. As we can see Fig. 2.7 is the culmination of all previous sections. All of the methods in this thesis use architectures that usually include modules of CNNs, pooling, and then ReLU. Multiple of these CNN, ReLU, and pooling modules are stacked and the output of the final module is flattened and put through a fully connected neural network. The final layer of the neural network is represented by a matrix that is $K \times M$, where K represents the number of classification classes and M is the information vector length, which is predefined.

2.9 Cross-Entropy Loss

Now that we've established the architecture of our models we now define the function we optimize. Typically for image classification in computer vision, the goal is for the model to output a probability that the image is a certain class. For example, if you train a model to classify dogs and cats it will output a probability of being a cat or dog for each image. Typically to train a vision model you require an annotated data set, which is a data set of images where each image's class is given. The standard objective function for image classification is **Cross-Entropy**, which can be defined as:

$$\mathbf{L} = -\sum_{i=1}^{m} y_i log(\hat{y}_i) \tag{2.3}$$

where m represents the number of classes and y_i is typically an identity function. y_i is set to 1 if i is the correct class label and 0 otherwise. $\hat{y_i}$ represents the probability of class i that the model has predicted. Since the final layer of vision models are typically a Fully Connected Neural Network we can rewrite Eq. 2.3 as:

$$\mathbf{L} = -\sum_{i=1}^{m} y_i log(f(x, w)) \tag{2.4}$$

where x is the feature mapping from the CNNs and w is the weights of the fully connected neural network. Typically the final output of the Fully Connected Neural Network has a **Softmax** applied, which is defined as:

$$\frac{e^{\hat{y_i}}}{\sum_{j=1}^{K} e^{\hat{y_j}}} \tag{2.5}$$

where $\hat{y_i}$ is the output from the Fully Connected Neural Network for class i. The Softmax function ensures that the outputs of the neural network obey the fundamental law of probability, which is that the sum of the probabilities within an event space equals 1.

2.10 Resnet

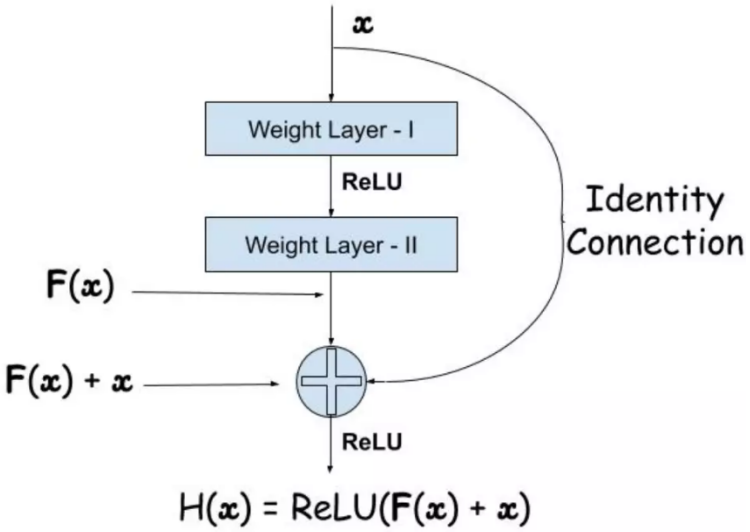


Figure 2.8: Flow of data through Resnet module represented by black arrows. x is put through a weighted layer and then summed with the output if $F(x)$. Image is taken from (59).

As mentioned in Sec. 2.5, deep neural architectures typically lead to improved performance but suffer from the **Vanishing Gradient** problem. Since the gradient values

are calculated by taking the product of each layer's gradient, multiplying many small values can lead to small gradients ineffective for learning or in extreme cases values too small to calculate, thus making learning impossible. Introduced in (28), Resnets was designed to circumvent this problem specifically for large vision models. As we can see in Fig. 2.8 the value x from the previous layers is added with the output of the current layer $F(x)$ and put through the next layer after being put through a ReLU layer. All methods in chapter 5 use Resnet for their backbone.

2.11 Reinforcement Learning (RL)

RL is a sub-field of Machine Learning where an **Agent** learns a **Policy** - a mapping from states to executable actions - that maximizes a numerical **Reward Signal** when interacting with an environment. **Observations** denote the current environment state. An example would be a robot (agent) learning how to play soccer. The environment typically encapsulates what the agent can observe, in this case, a soccer field, goals posts, and ball, (observations) and may include executable actions (pass, dribble, shoot, move) and the reward signal would be how many points the agent has scored. Reward signals may be hand-crafted in order to encourage the agent to learn desirable behaviors by giving a positive numerical value when an action leads to a desirable result is performed and punish it for performing undesired behaviors (the opposing agent scoring) where the agent is penalized with a negative reward, hence the term "reinforcement."

2.12 Markov Decision Process

A nice way to present RL problems is in **Markov Decision Process** (MDP) form. MDP is a discrete-time stochastic control process. Environment states are represented as discrete "points of time" where an agent affects state variables with each action sending the agent to the same or new state with some probability. A key property of MDPs is the **Markov Property**, which imposes that all relevant information for taking the next action is encapsulated into the current state and hence does not depend on knowledge of previous states or actions. We primarily consider finite time horizon MDPs in this work, which are tasks with a finite number of discrete time steps. A sequence of interactions that end in termination is typically referred to as an **Episode**. The sequence of state-action pairs observed via exploration using a policy is called a **Trajectory**. An example

of a basic MDP can be seen in Fig. 2.9. Fig. 2.9 represents a person's state and what action they can take. We see the three actions are: sleep, ice cream, and run. The red lines represent the probability of taking an action and ending up in that state. For example, after having ice cream there is a 10% chance the agent has ice cream again. Notice the numbers of all arrows going outward from a node in Fig. 2.9 sum to 1, this is to satisfy the **Law Of Total Probability**, which is the sum of possible events should sum to 1.

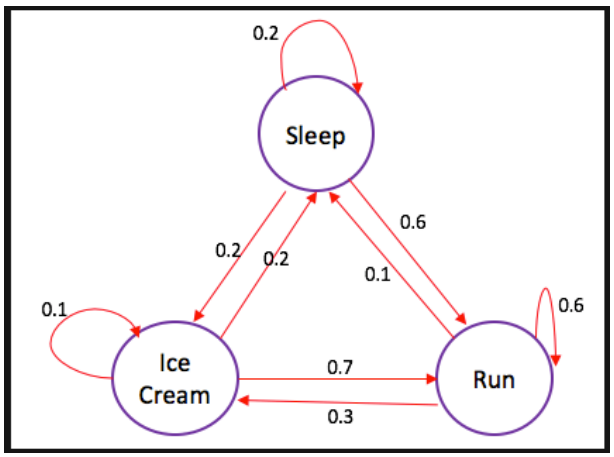


Figure 2.9: Example of an MDP. Three states are represented by purple circles: Sleep, Ice Cream, and Run. Transition probabilities are represented with a red line and an arrow from the previous state to the next state with a number representing the probability of transition. Image taken from (9)

2.13 RL to MDP

MDPs serve as a nice formulation for RL and offers useful theoretical benefits. Let $\mathcal{S}$ and $\mathcal{A}$ denote the state and action space. We can define $p(s_{t+1}|s_t, a_t), s_{t+1}, s_t \in \mathcal{S}, a_t \in \mathcal{A}$ as the probability of getting to state s_{t+1} from state s_t after taking action a_t, where $t \in \mathcal{T}$ denotes the time step and $\mathcal{T}$ is a discrete range of integers. $p(s_{t+1}|s_t, a_t)$ is assumed to maintain the Markov Property mentioned in the previous section. We can define $\mathcal{R}(s_t, a_t)$ as the reward function, which is the reward you receive at $t+1$ for taking an action at a state. The reward function may be deterministic or stochastic and can be dense or sparse.

Image-based RL is more reflective of a **Partially Observable MDP**(POMDP) (74) as we are only given images of the agent's current state as an RGB image. In order to shape it closer to a typical MDP a technique called **Frame Stacking** is employed. Multiple RGB observations are stacked representing the current state and the previous

k where k represents the number of frames to stack and is usually a predetermined hyper-parameter thus making $s_t \in R^{3k \times h \times w}$. The first dimension represents RGB image multiplied by a number of stacks then the second dimension is the image height h and the third is the image width w.

2.14 Bellman Optimality Equations

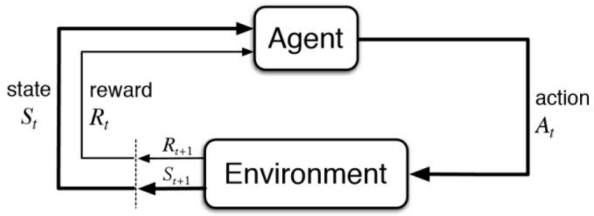


Figure 2.10: Image of agent and environment interaction. t represents a discrete time step. Image is taken from (8).

In RL the goal is to maximize **Expected Rewards**, which can be defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.6}$$

$\gamma \in [0, 1]$ represents a discount applied to future rewards as typically immediate rewards are more valuable. k represents a discrete time step between the present and future. t represents the current time step. $R_i, i \in [t, \infty]$ represents the amount of reward received at time i.

The **Q-Value Function**(62) or "quality value function" determines how good an action is taken from a state s under a policy π.

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] = \sum_{s' \in S} P(s'|s, a)[r(s, a, s') + \gamma v_\pi(s')] \tag{2.7}$$

where G_t is defined in Eq. 2.6. A_t is action and S_t represents the state at time t. $r(s, a, s')$ is the reward received for taking action a at state s and arriving at state s'. As we can see the Q-value function is recursive by nature as it requires the value function, which is defined in Eq. 2.8 of the next state s'. The Q-Value function outputs how good it is to take an action a in state s, following the policy π.

Next, we define **Value Function**(62), which is a measure of how good it is for the

agent to be in a state. Which we define as.

$$v_\pi(s) = E_\pi[G_t|S_t = s] = \sum_{a \in A} q_\pi(s,a)\pi(a|s) \tag{2.8}$$

As we can see the value function relies on the Q-value function. What the value function calculates is how valuable it is to be in state s, under policy π_θ. Since π is typically a distribution over actions it's the probability of taking action a multiplied by the Q-value.

The **Optimal Policy** is defined as:

$$\pi \geq \pi' \text{ if and only if } v_\pi(s) \geq v_{\pi'}(s), \forall s \in S \tag{2.9}$$

Eq. 2.9 is essentially saying that a policy π is relationally better than or equal to another π' on the condition that the value for every state under the policy is better. This leads to the **Optimal State-Value Function** and is shown in Eq. 2.10 and the **Optimal Action-Value Function** as shown in Eq. 2.11.

$$v_*(s) = \max_{a \in A} q_*(s,a) \tag{2.10}$$

$$q_*(s,a) = \sum_{s' \in S} r(s,a,s') + \gamma \sum_{s' \in S} P(s'|s,a)V_*(s') \tag{2.11}$$

Eq. 2.11 and Eq. 2.10 make up the **Bellman Optimality Equations**

2.15 Mathematical Derivation Of The Policy Gradient

We will leverage the definitions in Sec. 2.14 to learn how to find the optimal policy. An agent is trying to maximize **Returns**, which is the cumulative reward by a **Trajectory** which is a combination of states and actions taken to get to those states under a policy. Environments may be stochastic; therefore to account for this, the policies are learned to maximize the **Expected Reward** during an episode, which is a finite number of steps T. This can be written as:

$$\sum_{t=0}^{T-1} E_{\tau \sim \pi_\theta}[r_{t+1}] \tag{2.12}$$

Where π represents the policy function, θ represents learned parameters of the

policy function, r_t is the reward given at time t, and τ is the trajectory. An important question then is how do we learn π_θ to maximize returns?

To learn a policy that maximizes rewards, methods in this thesis leverage what is known as **Off-policy Policy Gradient** methods, which involve storing previous interactions with the environment into a **Replay Buffer**, typically a state, action, reward, and next state are stored. Off-policy RL methods can in theory learn from experiences stored using any policy. **Experience Replay** is a memory storage technique for storing the previous episodes. We also introduce $\beta(a|s)$, which is the **Behavior Policy** or the policy used to collect samples, versus the aforementioned π_θ which represents the agent's policy. Thus the objective function of off-policy methods becomes:

$$\begin{aligned} \mathcal{J}(\theta) &= \sum_{s \in S} d^\beta(s) \sum_{a \in A} q_{\pi_\theta}(s,a)\pi_\theta(a|s) \\ &= E_{s \sim d^\beta}[\sum_{a \in A} q_{\pi_\theta}(s,a)\pi_\theta(a|s)] \end{aligned} \tag{2.13}$$

Where $d^\beta(s)$ represents the stationary distribution of the behavior policy β. As we can see Eq. 2.13 stems from 2.8. $s \sim d^\beta$ simply means that the states are sampled by following the stationary distribution of the behavior policy. The gradient can be derived as:

$$\begin{aligned} \Delta_\theta \mathcal{J}(\theta) &= \Delta_\theta E_{s \sim d^\beta}[\sum_{a \in A} q_{\pi_\theta}(s,a)\pi_\theta(a|s)] \\ &= E_{s \sim d^\beta}[\sum_{a \in A} (q_{\pi_\theta}(s,a)\Delta_\theta \pi_\theta(a|s) + \pi_\theta(a|s)\Delta_\theta q_{\pi_\theta}(s,a))] \\ &\approx E_{s \sim d^\beta}[\sum_{a \in A} (q_{\pi_\theta}(s,a)\Delta_\theta \pi_\theta(a|s)] \\ &= E_{s \sim d^\beta}[\sum_{a \in A} \beta(a|s) \frac{\pi_\theta(a|s)}{\beta(a|s)} q_{\pi_\theta}(s,a) \frac{\Delta_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)}] \\ &= E_\beta[\sum_{a \in A} \frac{\pi_\theta(a|s)}{\beta(a|s)} q_{\pi_\theta}(s,a) \Delta_\theta ln \pi_\theta(a|s)] \end{aligned} \tag{2.14}$$

$\frac{\pi_\theta(a|s)}{\beta(a|s)}$ in Eq. 2.14 represents the importance weight. $\Delta_\theta q_{\pi_\theta}$ is very hard to calculate in reality. That is why we ignore it in Eq. 2.14 and get an approximation. This has been shown to be sufficient for improving the policy (15).

2.16 Soft Actor-Critic

We will begin by defining **Actor-Critic** methods (35). As noted in Sec. 2.14 two equations rely on each other's definitions for calculating how good a policy is: There is the Value function Eq. 2.8 and the Q-value function Eq. 2.7. Actor-Critic methods disentangle the responsibility of learning the value of a policy and the policy itself between an **Actor** and a **Critic** respectively. The Critic estimates the value of the current policy, while the Actor updates the policy based on what the Critic assesses as the expected reward. This is what is known as **Policy Iteration**, a 2-step process alternating between policy evaluation and policy improvement. To understand this 2-step process we define the **Bellman Operator**, which is:

$$[\mathcal{T}_\pi v_\pi](s) = E_{a\sim\pi(\cdot|s)}[r(s,a) + \gamma E_{s'|s,a}[v_\pi(s')]] \tag{2.15}$$

$r(s,a)$ is the reward received by taking action a at state s. What the Bellman operator is calculating is the expected reward from sampling from a policy.

Soft Actor-Critic (23) introduces **Soft Policy Iteration**(23), is similar to Policy Iteration but leverages slightly modified versions of the 2-step Policy Iteration equations. **Soft Bellman Operator**(69) which is defined as:

$$q_\pi^{soft}(s,a) = r(s,a) + \gamma E_{a'\sim\pi}[q_\pi^{soft}(s',a') - \alpha log\pi(a'|s)] \tag{2.16}$$

α denotes a scaling factor. The key thing to note in the Eq. 2.16 is $E_{a'\sim\pi}[-log\pi(a'|s)]$, which follows from the definition of **Entropy**:

$$H(x) = E[-logp(x)] \tag{2.17}$$

where $p(x)$ is a distribution. Entropy is defined as the average amount of information on a random variable. It has been used in Information Theory to quantify uncertainty. If the agent converges too quickly to a policy (being confident), there may be other unexplored policies that are superior. On the other hand, too much exploration (being uncertain) means not improving upon the best policy currently found. By adding entropy to the Q-value function we reward the agent for uncertainty. α determines how important

exploration should be. This is how **Soft Policy Evaluation** is done.

Now we examine **Soft Policy Iteration**(23), which is done using **Kullback-Leibler Divergence**, denoted by $KL(f||g)$. We define the policy iteration formula as:

$$E[KL(\pi(\cdot|s)||\frac{exp(q_\pi(s,\cdot))}{\sum_a exp(q_\pi(s,a))})] \tag{2.18}$$

Where $KL(f||g) = \sum_{x \in X} f(x) log(\frac{f(x)}{g(x)})$, which calculates the weighted difference in information between two distributions f and g at each point x. Eq. 2.18 improves the policy by adjusting the policy distribution for a state s to be more confident or more peaked around the action that gives the best state-action value. Typically the critic and actor networks are neural networks as described in section 2.1. This thesis only concerns itself with image-based RL so it leverages the model described in Sec. 2.8. Typically the agent is represented by a Gaussian distribution when the action space is continuous and a Categorical distribution when the action space is discrete.

Chapter 3 Related Works

In this chapter, we described the related works required to understand chapters 5 and 6. Sec. 3.1 defines calibration and methods to evaluate it, 3.2 and 3.3 describe the theoretical intuitions motivating the disentangled training and norm regularization used in methods described in Ch. 5. Sec. 3.4 gives a brief overview of augmentations and their effects on RL as a preface for Ch. 6.

3.1 Calibration Metrics

Calibration focuses on improving the predictive probability distribution such that it's similar to the distribution of classes in the data. For example, if we have a positive and negative class in our training data and 7 out of 10 are positive. We would expect the average predicted probability of the positive class to be 70%. Calibration methods aim to improve this with a held-out data set as typically models have good calibration to the data they are trained on, which may not be reflective of its calibration to unseen data.

We Assume a data point $X_i \in \mathbf{X}, i \in [1, N]$ each associated with a label $Y \in \mathbf{Y} = \{1, ..., K\}$. We would like our model M where $M(X_i) = (\hat{Y}_i, \hat{P}_i)$ where $\hat{Y}_i$ is the class prediction and $\hat{P}$ is the probability/confidence given by the model to be close to the ground truth distribution $P(Y_i|X_i)$. Ideally, $\hat{P}_i$ is well calibrated which means that it represents the likelihood of the true event $\hat{Y}_i = Y_i$. *Perfect calibration* (22) can be defined as:

$$\boldsymbol{P}(\hat{Y}_i = Y_i | \hat{P}_i = P_i) = P_i, \forall P_i \in [0, 1] \tag{3.1}$$

Ways of evaluating Calibration are as follows:

Expected Calibration Error (ECE)

Expected Calibration Error (50) evaluates calibration by calculating the difference in expectation between the confidence and accuracy or:

$$\sum_{m=1}^{M} \frac{|B_m|}{n} |acc(B_m) - conf(B_m)| \tag{3.2}$$

$\hat{P}$ is the confidence estimate, $\hat{Y}$ is the class prediction and p is the percent correctly classified as class $\hat{Y}$. This can also be computed as the weighted average of bins' accuracy/confidence difference:

$$\textbf{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |accuracy(B_m) - confidence(B_m)| \tag{3.3}$$

where n is the total number of samples. Perfect calibration is achieved when bins B_m confidence equals accuracy and ECE $= 0$. "Bins" in the context of ECE is the set of classes and each data point predicted on by the model is put into the respective bin representing the class predicted by the model.

Negative Log Likelihood (NLL)

A way to measure a model's probabilistic quality is to use Negative Log Likelihood (27). Given a probabilistic model $P(Y|X)$ and N samples it is defined as:

$$\textbf{L} = -\sum_{i=1}^{N} log(\hat{P}(Y_i|X_i)) \tag{3.4}$$

where $\hat{P}$ is the predicted distribution of the ground truth P and Y_i is the true label for input X_i. NLL belongs to a class of strictly proper scoring rules (21). A scoring rule is strictly proper if it is uniquely optimized by only the true distribution. NLL is the negative of the logarithm of the probability of the true outcome. If the true class is assigned a probability of 1, NLL will be minimum with value 0.

Brier

The Brier score (11) measures the accuracy of probabilistic predictions. Across all predicted items N in a set of predictions, the Brier score measures the mean squared difference between the predicted probability assigned to the possible outcome for $i \in [1, N]$ and the actual outcome.

$$\textbf{BS} = (1/N) \sum_{t=1}^{N} \sum_{i=1}^{R} (f_{ti} - o_{ti})^2 \tag{3.5}$$

where R is the number of possible classes, N is the overall number of instances of all classes. f_{ti} is the approximated probability of the forecast o_{ti} in one hot encoding. Brier score can be intuitively decomposed into three components: uncertainty, reliability, and resolution (49), and it is also a proper scoring rule.

3.2 Angular Visual Hardness

Prior works(13) have explored where most of the predictive power of CNNs stems from, this section elaborates on key points in that work related to this thesis as intuition derived from (13) are leveraged in Ch. 5. In Sec. 2.8 we described a standard CNN model. An image is passed through a CNN g and outputs a vector. That vector is passed to a linear layer represented as a matrix where each row represents a class vector as described in Sec. 2.8. For brevity, we'll assume it's a single linear layer. We can define each row in the final linear layer as $\vec{w_i}, i \in [1, K]$. To calculate the probability that the input x is class i, we use Eq. 5.1 and can be written as:

$$\frac{e^{g(\vec{x}) \cdot \vec{w_i}}}{\sum_{i=1}^{K} e^{g(\vec{x}) \cdot \vec{w_i}}} \tag{3.6}$$

The key thing to note is that $g(\vec{x})$ and $\vec{w_i}$ are vectors therefore the dot product is taken. The dot product from a Euclidean perspective can be derived as $\vec{w_i} \cdot g(\vec{x}) = \|\vec{w_i}\| \|g(\vec{x})\| cos(\theta)$.

(13) introduced **Angular Visual Hardness**(AVH), which is defined as:

$$\mathcal{AVH}(x) = \frac{\mathcal{A}(g(x), w_y)}{\sum_{i=1}^{K} \mathcal{A}(g(x), w_i)} \tag{3.7}$$

where $\mathcal{A}(u, v) = arccos(\frac{u \cdot v}{\|u\|\|v\|})$. What $\mathcal{A}()$ calculates is the angle between the class vector and the embedded vector of the input. AVH represents a score of the angular similarity of the information vector of the input and a class vector about its angular similarity with all other class vectors. (13) make the following observations about AVH:

- The primary determinant of class assignment is AVH
- Norm of input embedding determines the confidence
- The norm of the input embedding is not bounded and therefore increases to infinity during training

3.3 Introduction to Temperature Scaling

Methods in Ch. 5 leverage a form of Temperature Scaling for improving the calibration of models. Temperature scaling is a simple form of Platt scaling (55). Temperature scaling uses a scalar T to adjust the confidence of the softmax probability in a classification model. Following the notation from the main paper, let l denote the logits. The temperature scalar is applied to all classes as follows:

$$P(y|x) = \frac{\exp \frac{1}{T} l_y}{\sum_{j=1}^{c} \exp \frac{1}{T} l_j} = \frac{\exp \left(\|\mathbf{w_y}\|_2 \frac{1}{T} \|\mathbf{x}\|_2 \cos \phi_y \right)}{\sum_{j=1}^{c} \exp \left(\|\mathbf{w_j}\|_2 \frac{1}{T} \|\mathbf{x}\|_2 \cos \phi_j \right)} \tag{3.8}$$

As described in Fig. 5.1a, the temperature effectively changes the slope of $\|\mathbf{x}\|_2$ from 1 to $\frac{1}{T}$. The temperature parameter is optimized by minimizing negative log-likelihood on a validation set while freezing all the other model parameters (22). Temperature scaling calibrates a model's confidence on in-distribution (IND) data and does not change accuracy. However, it does not provide any mechanism to improve calibration on shifted distribution and is inferior to other uncertainty estimation methods in terms of calibration (52).

3.4 Augmentation in RL

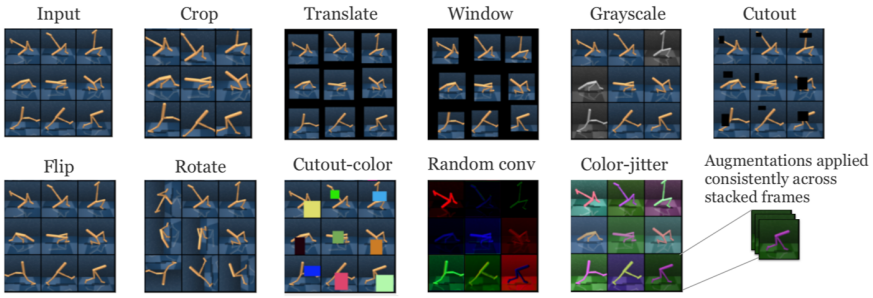


Figure 3.1: Examples of different augmentations applied to Deep Mind Control Suite images(63). Image is taken from (38).

Now that we've established the necessary foundations in the previous sections for calibration in CV, we now establish the necessary background of what is known about the effects of image augmentation in RL. Image augmentation has been shown to help the CV model learn more robust representations (53; 47). The same philosophy has been applied to pixel-based RL with interesting results (38; 68). What was interesting was that, unlike CV where a set of augmentations tends to help models have a more robust representation of different classes, RL seemed to benefit from specific augmentations and applying only one throughout training (38). The utility of such augmentations is improved sample efficiency or convergence to the best achievable policy faster. There is no universal augmentation that helps sample efficiency across all problem domains, but there do exist generally applicable ones, namely: shift and crop. An example of a crop can be seen in fig. 3.1. Shift (68) applies padding to an image and then crops back to the original dimensions. It has been shown that augmentations counter-act the inherent high-frequency bias that CNNs have (19) and allow agents to learn visual features relevant to the task (12).

The dangers of using augmentations are that some augmentations termed **Strong Augmentations** have shown to be detrimental to policy learning (38). The utility of strong augmentation is their visual similarity to common visual perturbations such as color change. The main issue is leveraging strong augmentations without impeding policy learning. In Ch. 6 we focus on how to strike a good balance between policy learning with augmentations that are known to regularize networks and strong augmentations which are visually reflective of distribution shift.

Chapter 4 Benchmarks For Evaluating Robustness

4.1 Cifar Image Dataset

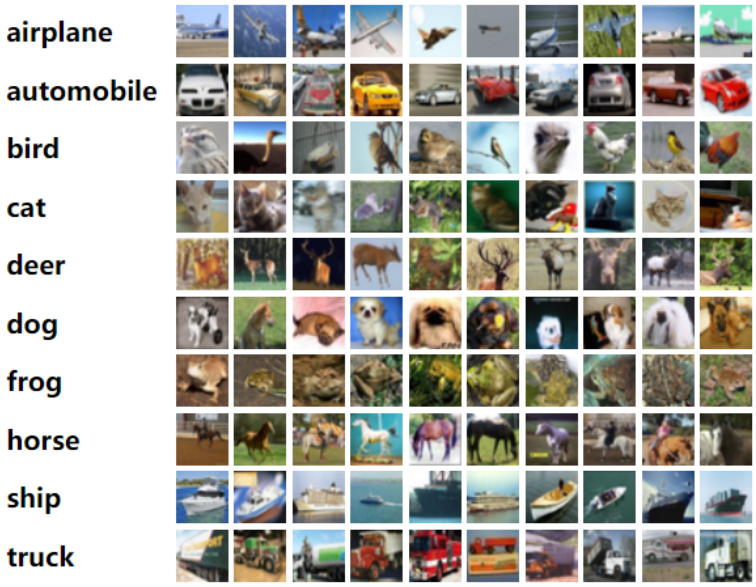


Figure 4.1: Example images from the Cifar 10 dataset. Image is taken from (2).

For CV-related tasks, we train our models on Cifar 10 and 100, both containing 60000 annotated images with 10 classes comprising animals and vehicles in Cifar 10 and 100 for Cifar 100. (29) offers code to apply corruptions to images of differing levels of severity from 1 to 5 an example of the different types of noises is shown in fig. 4.2. In order to test CV model calibration in the face of noise we use Cifar 10 and 100 Corrupt(29). Examples of images corrupted by (29) can be seen in Fig. 4.2 and 4.3. A standard method to evaluate calibration is to train on Cifar 10 and 100 and evaluate the confidence and accuracy correlation using Cifar 10 and 100 Corrupt.

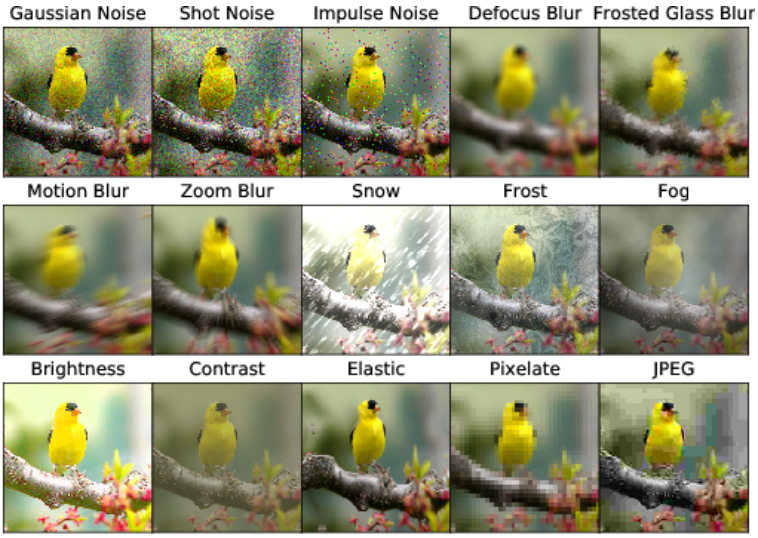


Figure 4.2: Example images of different types of noises applied to images from (29). Image is taken from (29).

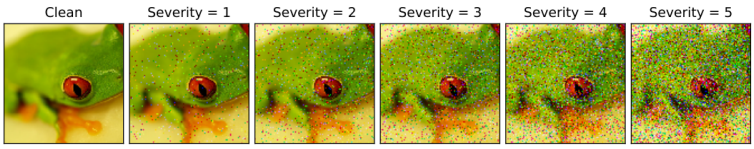


Figure 4.3: Example differing severity of noise application using (29). Image is taken from (29).

4.2 Deep Mind Control Suite Generalization Benchmark

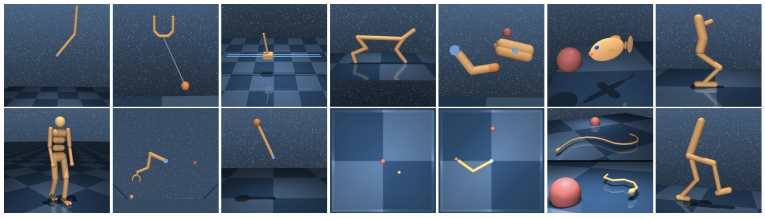


Figure 4.4: Samples of a single frame from tasks in Deep Mind Control Suite. Image is taken from (63).

We assess all RL models on Deep Mind Control Suite (DMC) (63). DMC consists of a large set of robotics tasks where the action is vectors representing continuous values that control different parts of the agent. As seen in fig. 4.4 the agents take on varying forms and DMC also offers a large variety of tasks from making the agent walk to catching a ball in a cup. Typically an agent's performance is evaluated on the cumulative reward it earned, based on the task. To evaluate generalization ability we benchmark on DMC Generalization Benchmark (DMC-GB). DMC-GB offers 4 modes: color easy, color

hard, video easy and video hard. The color modes randomly change the color of the agent, flooring, and background. The video settings randomly change the background to another image. We have a table containing task descriptions used to benchmark methods in this paper in Tab. 4.1

Table 4.1: Table containing: action space dimension, a brief description of the task, and if rewards are dense. Descriptions are taken from (25).

Domain, Task	Description	Action Vector Size	Dense Rewards
Walker, Walk	A planar walker that is rewarded for walking forward at a target velocity.	6	Yes
Walker, Stand	A planar walker that is rewarded for standing with an upright torso at a constant minimum height.	6	Yes
Cartpole, Swingup	Swing up and balance a pole by applying forces to a cart at its base. The agent is rewarded for balancing the pole within a fixed threshold angle.	1	Yes
Ball In Cup, Catch	An actuated planar receptacle is to swing and catch a ball attached by a string to its bottom	2	No
Finger, Spin	A manipulation problem with a planar 3 DoF finger. The task is to continually spin a free body	2	No

Chapter 5 Geometric Sensitivity Decomposition

During development, deep learning models are trained and validated on data from the same distribution. However, in the real world sensors degrade and weather conditions change. Similarly, subtle changes in image acquisition and processing can also lead to a distribution shift of the input data. This is often known as *covariate shift*, and will typically decrease the performance (e.g. classification accuracy). However, it has been empirically found that the model's *confidence* remains high even when accuracy has degraded (29). The process of aligning confidence to empirical accuracy is called model *calibration*. Calibrated probability provides valuable uncertainty information for decision-making. For example, knowing when a decision cannot be trusted and more data is needed is important for safety and efficiency in real-world applications such as self-driving (10) and active learning (67).

A comprehensive comparison of calibration methods has been studied for in-distribution (IND) data (22), However, these methods lead to unsatisfactory performance under distribution shift (52). To resolve the problem, high-quality uncertainty estimation (34; 52) is required. Principled Bayesian methods (18) model uncertainty directly but are computationally heavy. Recent deterministic methods (65; 43) propose to improve a model's *sensitivity* to input changes by regularizing the model's intermediate layers. In this context, sensitivity is defined as preserving distance between two different input samples through layers of the model. We would like to utilize the improved sensitivity to better detect Out-of-Distribution (OOD) data. However, these methods introduce added architecture changes and large combinatorics of hyperparameters.

Unlike existing works, we propose to study sensitivity from a geometric perspective. The last linear layer in a softmax-linear model can be decomposed into the multiplication of a norm and a cosine similarity term (44; 13; 33; 45). Geometrically, the angular similarity dictates the membership of input, and the norm only affects the confidence in a softmax-linear model. Counter-intuitively, the norm of a sample's feature embedding

exhibits little correlation to the hardness of the input (13). Based on this observation, we explore two questions: 1) why is a model's confidence insensitive to distribution shift? 2) how do we improve model sensitivity and calibration?

We hypothesize that in part an insensitive norm is responsible for bad calibration, especially on shifted data. We observe that the sensitivity of the angular similarity increases with training whereas the sensitivity of the norm remains low. More importantly, calibration worsens during the period when the norm increases while the angular similarity changes slowly. This shows a concrete example of the inability of the norm to *adapt* when accuracy has dropped. Intuitively, training on clean datasets encourages neural networks to *always* output increasingly large feature norms to continuously minimize the training loss. Because the probability of the prevalent class of input is proportional to its norm, larger norms lead to smaller training loss when most training data have been classified correctly (See Sec. 5.1.1). This renders the norm insensitive to input differences because the model is trained to *always* output features with the large norm on clean data. While we have put forth that the norm is poorly calibrated, we must emphasize that it can still play an important role in model calibration (See Sec. 5.2.1).

To encourage sensitivity, we propose to decompose the norm of a sample's feature embedding and the angular similarity into two components: *instance-dependent* and *instance-independent*. The instance-dependent component captures the sensitive information about the input while the instance-independent component represents the insensitive information serving solely to minimize the loss on the training dataset. Inspired by the decomposition, we analytically derive a simple extension to the current softmax-linear model, which learns to disentangle the two components during training. We show that our model outperforms other deterministic methods (despite their significant complexity) and is comparable to multi-pass methods with fewer training hyperparameters in Sec. 5.2.1.

In summary, our contributions are fourfold:

- We study the problem of calibration geometrically and identify that the insensitive norm is responsible for bad calibration under distribution shift.
- We derive a principled but simple geometric decomposition that decomposes the norm into an instance-dependent and instance-independent component.

- Based on the decomposition, we propose a simple training and inference scheme to encourage the norm to reflect distribution changes.
- We achieve state-of-the-art results in calibration metrics in the face of corruption while having arguably the simplest calibration method to implement.

This work was done in collaboration with Junjiao Tian and Yen-Chang Hsu and was accepted into Neurips 2021 and won the Spotlight Paper award.

5.1 Method

Following our hypothesis that the insensitivity of the norm is responsible for bad calibration on distribution-shifted data, we propose geometric sensitivity decomposition (GSD) for the norm. We first introduce the geometric perspective of the last linear layer in Sec. 5.1.1 and then derive GSD in Sec. 5.1.2. To improve the sensitivity of the norm and model calibration on shifted data, we propose a GSD-inspired training and inference procedure in Sec. 5.1.3 and Sec. 5.1.4.

5.1.1 Norm and Similarity

The output layer of a neural network can be written as a dot-product $< \mathbf{x}, \mathbf{w_y} >$, where $\mathbf{x}$ is the embedded input and $\mathbf{w_y}$ is the weight vector associated with class y. Though seemingly simple there are strongly geometric and calibration-related intuitions drawn from this. Several prior works (44; 33; 13) have studied the effects decomposition of the last linear layer in a softmax model can have on classification. The output layer can be decomposed into angular similarity $\cos \phi_y$ and norm $\|\mathbf{x}\|_2$.

$$P(y|x) = \frac{\exp l_y}{\sum_{j=1}^{c} \exp l_j} = \frac{\exp \left(\|\mathbf{w_y}\|_2 \|\mathbf{x}\|_2 \cos \phi_y \right)}{\sum_{j=1}^{c} \exp \left(\|\mathbf{w_j}\|_2 \|\mathbf{x}\|_2 \cos \phi_j \right)} \tag{5.1}$$

where $\|\mathbf{w_y}\|_2$ is the norm of a specific classifier in the linear layer. We'll use this geometric view of the linear layer instead of the dot-product representation.

Based on this perspective, we base the foundation of our work on the following observations from prior works (44; 33; 13): 1) The probability/confidence of the prevalent class of input is proportional to its norm (33). 2) While the norm of a feature strongly scales the predictive probability, due to its unregularized nature the norm is not sensitive to the hardness of the input (13). In other words, the norm could be the reason for the

bad sensitivity of the confidence to input distribution shift. Consequently, the insensitive norm can be causally related to bad calibration. We will examine a strong correlation between the quality of calibration and the magnitude of the norm in Sec. 5.2.2.

5.1.2 Geometric Sensitivity Decomposition of Norm and Angular Similarity

To motivate the subsequent geometric decomposition, we can revisit the softmax model, $P(y|x) \propto \exp\left(\|\mathbf{w_y}\|_2 \|\mathbf{x}\|_2 \cos\phi_y\right)$. Three terms are contributing to the magnitude of the exponential function, $\|\mathbf{w_y}\|_2$, $\|\mathbf{x}\|_2$ and $\cos\phi_y$. Due to weight regularizations, $\|\mathbf{w_y}\|_2$ is most likely very small, while $\cos\phi_y \in [-1, 1]$. Therefore, the only way to obtain a high probability/confidence on training data and minimize cross-entropy loss is to 1) push the norm $\|\mathbf{x}\|_2$ to a large value and 2) keep $\cos|\phi_y|$ of the ground truth class close to one, i.e., $|\phi_y|$ close to zero. This is further supported by (60), where it was shown that logits of the ground truth class must diverge to infinity to minimize cross-entropy loss under gradient descent. In this process, models tend towards *large* norms and *small* angles for all training samples.

Therefore, we propose to *decompose* the norms of features into two components: an *instance-independent* scalar offset and an *instance-dependent* variance factor, which we define in Eq. 5.2. The role of the instance-independent offset $\mathcal{C}_x$ is to minimize the loss on the entire training set and the instance-dependent component Δx accounts for differences in samples. Therefore, if we can disentangle the instance-independent component from the instance-dependent component, we can obtain a norm that is sensitive to the hardness of data. Following this logic, we decompose the norm into two components.

$$\|\mathbf{x}\|_2 = \|\Delta x\|_2 + \mathcal{C}_x \tag{5.2}$$

Similarly, we relax the angles such that the predicted angular similarity does not need to be close to one on the training data, i.e., making the angles larger. To achieve this, we introduce an instance-independent relaxation angle $\mathcal{C}_\phi$ and an instance-dependent angle $\Delta\phi_y$. Analogous to the norm decomposition, the scalar $\mathcal{C}_\phi$ serves solely to minimize the training loss while the instance-dependent $\Delta\phi_y$ accounts for differences in samples.

Because we need to account for the sign of the angle, we put an absolute value on it.

$$|\phi_y| = |\Delta\phi_y| - |\mathcal{C}_\phi| \tag{5.3}$$

The $\|\Delta\mathbf{x}\|_2, |\Delta\phi_y|$ are the instance-dependent components and $\mathcal{C}_x, |\mathcal{C}_\phi|$ are the instance-independent components. We can rewrite the pre-softmax logits in Eq. 5.1 with the decomposed norm and angular similarity. (Detailed derivation in Sec. 5.1.5 in the Appendix.)

$$\begin{aligned}\|\mathbf{x}\|_2 \cos\phi_y &= \|\mathbf{x}\|_2 \cos|\phi_y| = (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\cos(|\Delta\phi_y| - |\mathcal{C}_\phi|) \\ &= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos|\mathcal{C}_\phi|}\cos|\Delta\phi_y| \times \left(1 - \sin|\mathcal{C}_\phi|^2\left(1 - \frac{\cos|\mathcal{C}_\phi|\sin|\Delta\phi_y|}{\sin|\mathcal{C}_\phi|\cos|\Delta\phi_y|}\right)\right)\end{aligned} \tag{5.4}$$

We can simplify the equation by **assuming** $\cos|\phi_y|$ **is close to one, which means** $|\phi_y|$ **is small.** This is because $|\phi_y|$ is the angle between the correct class weight and x, which means as training ensues, the angle converges to 0 and thus the cosine similarity converges to 1. (Please see Sec. 5.1.6 for empirical support.)

$$\frac{\cos|\mathcal{C}_\phi|\sin|\Delta\phi_y|}{\sin|\mathcal{C}_\phi|\cos|\Delta\phi_y|} = \frac{\sin(|\Delta\phi_y| + |\mathcal{C}_\phi|) + \sin|\phi_y|}{\sin(|\Delta\phi_y| + |\mathcal{C}_\phi|) - \sin|\phi_y|} \approx 1 \tag{5.5}$$

Therefore, Eq. 5.4, omitting the absolute value on angles because cos is an even function, simplifies:

$$\begin{aligned}\|\mathbf{x}\|_2 \cos\phi_y &\approx (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos\mathcal{C}_\phi}\cos\Delta\phi_y \\ &= \left(\frac{1}{\cos\mathcal{C}_\phi}\|\Delta\mathbf{x}\|_2 + \frac{1}{\cos\mathcal{C}_\phi}\mathcal{C}_x\right)\cos\Delta\phi_y \\ &= \left(\frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha}\right)\cos\Delta\phi_y\end{aligned} \tag{5.6}$$

Because $\cos\mathcal{C}_\phi$ and $\mathcal{C}_x$ are instance-independent, we denote them as α and β respectively. **This geometric decomposition of norm and cosine similarity inspires us to include** α **and** β **as free trainable parameters in a new network and the network can learn to predict the more input-sensitive** $\|\Delta\mathbf{x}\|_2$ **and** $\Delta\phi_y$ **instead of**

the original $\|\mathbf{x}\|_2$ and ϕ_y. While both the angle and norm can be decomposed we direct the focus to the norm as the angle is *already* calibrated to accuracy (13). In other words, angles have been shown to be sensitive to input changes in (13).

5.1.3 Disentangled Training

Following the derivation in Eq 5.6, we replace the norm, $\|\mathbf{x}\|_2$, in Eq. 5.1 by $\left(\frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha}\right)$ and ϕ_y by $\Delta\phi_y$. $\|\Delta\mathbf{x}\|_2$ and $\Delta\phi_y$ are now learned outputs from a new network instead as shown in Eq. 5.6:

$$P(y|x) = \frac{\exp l_y}{\sum_{j=1}^{c} \exp l_j} = \frac{\exp\left(\|\mathbf{w_y}\|_2 \left(\frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha}\right) \cos\Delta\phi_y\right)}{\sum_{j=1}^{c} \exp\left(\|\mathbf{w_j}\|_2 \left(\frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha}\right) \cos\Delta\phi_j\right)} \tag{5.7}$$

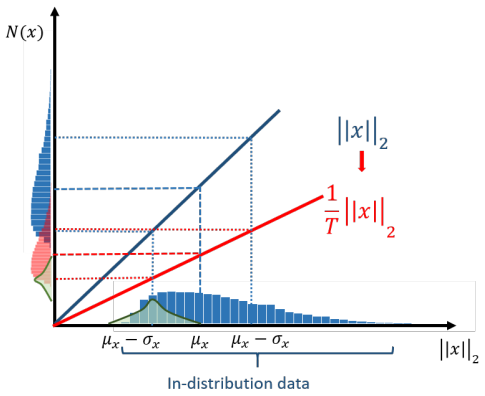
The new model can be trained using the same training procedures as the vanilla network without additional hyperparameter tuning, changing the architecture or extended training time. Even though the outputs of the new network, $\|\Delta\mathbf{x}\|_2$ and $\Delta\phi_y$, only approximate the original geometric relationships with Eq. 5.6, the effect of α and β reflects the decomposition in Eq. 5.3 and Eq. 5.2.

- β encodes an instance-independent scalar $\mathcal{C}_x$ of the norm. A larger β corresponds to a smaller instance-dependent component $\|\Delta\mathbf{x}\|_2$.
- α encodes the cosine of a relaxation angle $\mathcal{C}_\phi$. A larger $\arccos\alpha$ corresponds to a larger $\mathcal{C}_\phi$ and therefore a larger $\Delta\phi_j$.

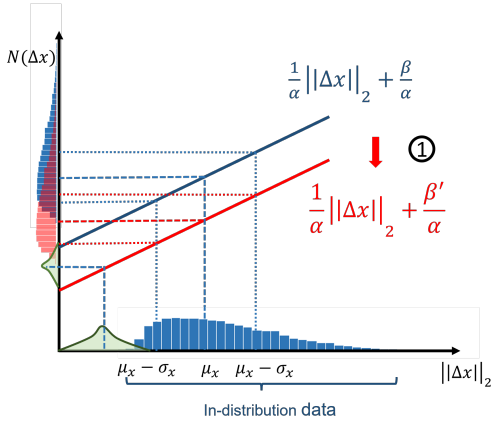
Because β encodes the independent component, the new feature norm $\|\Delta\mathbf{x}\|_2$ becomes sensitive to input changes and maps OOD data to lower norms than IND data as we can see in Fig. 5.3a, 5.3b. We regularize α such that the instance-independent component $\mathcal{C}_\phi$ is small. Specifically, we penalize $\|\alpha - 1\|_2^2$ because $\alpha = \cos\mathcal{C}_\phi$, i.e., if $\alpha \approx 1$, $\mathcal{C}_\phi \approx 0$. We empirically found that a larger relaxation angle $\mathcal{C}_\phi$ deteriorates performance because the angular similarity already correlates well with difficulty of data (13) and we do not need to encourage a large relaxation. Sec. 5.2.3 will empirically verify this.

5.1.4 Disentangled Inference

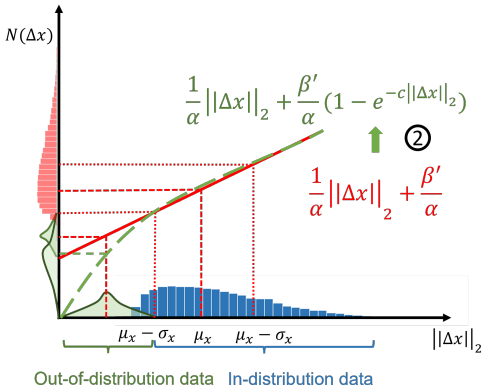
The decomposition theory in Sec. 5.1.2 provides a geometric perspective on the sensitivity of the norm and the angular similarity to input changes and inspires a disen-



(a) Temperature Scaling



(b) Ours: calibration Step 1



(c) Ours: Calibration Step 2

Figure 5.1: **Calibration Procedure** (a): Temperature Scaling (22) changes the slope of the effective norm based on in-distribution (IND) data (See 3.3)

tangled model in Sec. 5.1.3. The new model uses a learnable affine transformation on the norm $\|\Delta\mathbf{x}\|_2$. Let's denote the affine transformed norm as the *effective norm* $\mathcal{N}(\Delta\mathbf{x}) \doteq \frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha}$. However, the training only *separates* the sensitive components of the norm and angular similarity, the model can still be overconfident due to the existence of insensitive components. Therefore, we can improve calibration by modifying insensitive components, e.g., β in our case. We propose a two-step calibration procedure that combines **in-distribution calibration** (Fig. 5.1b) and **out-of-distribution detection** (Fig. 5.1c) based on two observations: 1) overconfident IND data can be easily calibrated on a validation set, similar to temperature scaling (22). 2) for OOD data, without access to a calibration set for OOD data, the best strategy is to map them far away from the IND data given that the model clearly distinguishes them.

The first step is calibrating the model on IND validation set (note our method does *not* rely on OOD validation data), similar to temperature calibration (22). However, instead of tuning a temperature parameter as shown in Fig. 5.1a, we simply tune the offset parameter β on the validation set in one of two ways: 1) grid-search based on minimizing Expected Calibration Error (see Sec. 5.2) 2) SGD optimization based on Negative Log Likelihood (22). Because these are post-training procedure, both methods are very efficient. We denote the new parameter as β'. As shown in Fig. 5.1b, by changing the offset, we decrease the magnitude of the norms after the affine transformation. Formally,

$$\mathcal{N}(\Delta\mathbf{x}) = \frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta}{\alpha} \rightarrow \mathcal{N}(\Delta\mathbf{x}) = \frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta'}{\alpha} \tag{5.8}$$

The second step approximates the calibrated affine mapping in Eq. 5.8 by a non-linear function which covers a wider range of the effective norm as shown in Eq. 5.9 and maps OOD data further away from IND data. Intuitively, when a sample is more likely IND, the non-linear function maps it closer to the calibrated transformation. When a sample is OOD, the non-linear function maps it more aggressively to a smaller magnitude, exponentially away from the IND samples.

$$\mathcal{N}(\Delta\mathbf{x}) = \frac{1}{\alpha}\|\Delta\mathbf{x}\|_2 + \frac{\beta'}{\alpha}(1 - e^{-c\|\Delta\mathbf{x}\|_2}) \tag{5.9}$$

where c is a hyperparameter which can be calculated as in Eq. 5.10. The non-linear function grows exponentially close to the calibrated affine mapping in Eq. 5.8 dictated by

$1 - e^{-c\|\mathbf{\Delta x}\|_2}$ as shown in 5.1c. Therefore, $e^{-c\|\Delta\mathbf{x}\|_2}$ can be viewed as an *error* term that quantifies how close the non-linear function is to the calibrated affine function in Eq. 5.8. Let μ_x and σ_x denote the mean and standard deviation of the distribution of the norm of IND sample embedding calculated on the validation set. We use the heuristic that when evaluated at one standard deviation below the mean, $\|\Delta\mathbf{x}\|_2 = \mu_x - \sigma_x$, the approximation error $e^{-c(\mu_x - \sigma_x)} = 0.1$. Even though the error threshold is a hyperparameter, using an error of 0.1 leads to state-of-the-art results across all models applied.

$$c = \frac{-ln(1 - error)}{\mu_x - \sigma_x} = \frac{-ln(0.9)}{\mu_x - \sigma_x} \tag{5.10}$$

In summary, the sensitive norm $\|\Delta\mathbf{x}\|_2$ is used both as a soft threshold for OOD detection and as a criterion for calibration. While similar post-processing calibration procedure exists, such as temperature scaling (22) (illustrated in Fig. 5.1a and further introduced in 3.3) it only provides good calibration on IND data and does not provide any mechanism to improve calibration on shifted data (52). Our calibration procedure can improve calibration on both IND and OOD data, without access to OOD data, because the training method extracts the sensitive component in a principled manner. Just as temperature scaling, the non-linear mapping needs only to be calculated *once* and adds no computation at inference.

5.1.5 Mathematical Derivation for Equation 5.4

We proposed to decompose the norm and angular similarity into instance-independent and dependent components.

$$\|\mathbf{x}\|_2 = \|\Delta x\|_2 + \mathcal{C}_x \tag{5.11}$$

$$|\phi_y| = |\Delta\phi_y| - |\mathcal{C}_\phi| \tag{5.12}$$

The $\|\Delta\mathbf{x}\|_2, |\Delta\phi_y|$ are the instance-dependent components and $\mathcal{C}_x, |\mathcal{C}_\phi|$ are the instance-independent components. We can rewrite the pre-softmax logits in Eq. 5.1 with the

decomposed norm and angular similarity.

$$
\begin{aligned}
\|\mathbf{x}\|_2 \cos\phi_y = \|\mathbf{x}\|_2 \cos|\phi_y| &= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\cos(|\Delta\phi_y| - |\mathcal{C}_\phi|) \\
&= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\left(\cos|\Delta\phi_y|\cos|\mathcal{C}_\phi| + \sin|\Delta\phi_y|\sin|\mathcal{C}_\phi|\right) \\
&= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos|\mathcal{C}_\phi|}\left(\cos|\Delta\phi_y|\cos|\mathcal{C}_\phi|^2 + \sin|\Delta\phi_y|\cos|\mathcal{C}_\phi|\sin|\mathcal{C}_\phi|\right) \\
&= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos|\mathcal{C}_\phi|}\cos|\Delta\phi_y|\left(\cos|\mathcal{C}_\phi|^2 + \cos|\mathcal{C}_\phi|\sin|\mathcal{C}_\phi|\frac{\sin|\Delta\phi_y|}{\cos|\Delta\phi_y|}\right) \\
&= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos|\mathcal{C}_\phi|}\cos|\Delta\phi_y|\left(\left(1-\sin|\mathcal{C}_\phi|^2\right) + \cos|\mathcal{C}_\phi|\sin|\mathcal{C}_\phi|\frac{\sin|\Delta\phi_y|}{\cos|\Delta\phi_y|}\right) \\
&= (\|\Delta\mathbf{x}\|_2 + \mathcal{C}_x)\frac{1}{\cos|\mathcal{C}_\phi|}\cos|\Delta\phi_y|\left(1-\sin|\mathcal{C}_\phi|^2\left(1-\underbrace{\frac{\cos|\mathcal{C}_\phi|\sin|\Delta\phi_y|}{\sin|\mathcal{C}_\phi|\cos|\Delta\phi_y|}}_{\approx 1 \text{ Eq. 5.14}}\right)\right) \\
&\approx \left(\frac{1}{\cos|\mathcal{C}_\phi|}\|\Delta\mathbf{x}\|_2 + \frac{\mathcal{C}_x}{\cos|\mathcal{C}_\phi|}\right)\cos|\Delta\phi_y|
\end{aligned}
\tag{5.13}
$$

We can simplify the equation by **assuming** $\cos|\phi_y|$ **is close to one, which means** $|\phi_y|$ **is small.** This is because $|\phi_y|$ is the angle between the correct class weight and x, which means as training ensues, the angle converges to 0 and thus the cosine similarity converges to 1. (Please see Sec. 5.1.6 for empirical support.)

$$
\frac{\cos|\mathcal{C}_\phi|\sin|\Delta\phi_y|}{\sin|\mathcal{C}_\phi|\cos|\Delta\phi_y|} = \frac{\sin(|\Delta\phi_y| + |\mathcal{C}_\phi|) + \sin|\phi_y|}{\sin(|\Delta\phi_y| + |\mathcal{C}_\phi|) - \sin|\phi_y|} \approx 1 \tag{5.14}
$$

5.1.6 Small Angle Assumption in Equation 5.5

Table 5.1: Average cosine similarity to the ground truth class on the training data set after training for 200 epochs

	CIFAR10			CIFAR100		
	ResNet-18	ResNet-34	ResNet-101	ResNet-18	ResNet-34	ResNet-101
$\cos\phi$	0.81	0.79	0.76	0.75	0.78	0.74

One reason for the small angle assumption in Eq. 5.5 is the observation that high-capacity models tend to be more miscalibrated (22) and our method is especially more effective in this case. When a model is sufficiently high-capacity compared to the diversity of the dataset, the assumption of small-angle is empirically more valid and the method can provide more significant improvement. All ResNet models are high-capacity deep models and the corresponding cosine similarity to the true class is close to one

during training as assumed in Sec. 5.1.2. Tab. 5.1 shows the average cosine similarity to the ground truth class on the training data.

5.2 Experiments

5.2.1 Experiments on Calibration

Table 5.2: **ResNet-28-10 on CIFAR10** averaged over 10 seed. † denotes results from (43). Our method outperforms other single-pass methods and is comparable to Deep Ensemble (37) on corrupted data. While the ensembled version of our model beats all multi-pass models.

	Method	Accuracy ↑		ECE ↓		NLL ↓	
		Clean	Corrupted	Clean	Corrupted	Clean	Corrupted
5*Single-Pass	Vanilla†	**96.0±0.01**	72.9±0.01	0.023±0.002	0.153±0.011	0.158±0.01	1.059±0.02
	DUQ†	94.7±0.02	71.6±0.02	0.034±0.002	0.183±0.011	0.239±0.02	1.348±0.01
	SNGP†	95.9±0.01	74.6±0.01	0.018±0.001	0.090±0.012	**0.138±0.01**	0.935±0.01
	Ours β' Grid-Searched	95.9±0.01	**74.9±0.05**	0.018±0.003	**0.067±0.010**	0.148±0.003	**0.826±0.03**
	Ours β' Optimized	95.9±0.01	**74.9±0.05**	**0.008±0.002**	0.085±0.012	0.140±0.004	0.853±0.04
3* Multi-Pass	Deep Ensembles†	96.6±0.01	**77.9±0.01**	0.010±0.001	0.087±0.004	0.114±0.01	0.815±0.01
	MC Dropout†	96.0±0.01	70.0±0.02	0.021±0.002	0.116±0.009	0.173±0.001	1.152±0.01
	Ours β'Grid-Searched	**96.62**	**77.9**	**0.007**	**0.069**	**0.108**	**0.773**

Table 5.3: **ResNet-28-10 on CIFAR100** averaged over 10 seeds. † denotes results from (43). Our method outperforms other single-pass methods and Deep Ensemble (37) on corrupted data. While the ensembled version of our model beats all multi-pass models

	Method†	Accuracy↑		ECE ↓		NLL ↓	
		Clean	Corrupted	Clean	Corrupted	Clean	Corrupted
5*Single-Pass	Vanilla†	79.8±0.02	**50.5±0.04**	0.085±0.004	0.239±0.020	0.872±0.01	2.756±0.03
	DUQ†	78.5±0.02	50.4±0.02	0.119±0.001	0.281±0.012	0.980±0.02	2.841±0.01
	SNGP†	**79.9±0.03**	49.0±0.02	**0.025±0.012**	0.117±0.014	0.847±0.01	2.626±0.01
	Ours β' Grid-Searched	79.8±0.03	49.8 ± 0.003	0.027±0.003	**0.081 ± 0.007**	0.787±0.009	**2.23±0.02**
	Ours β' Optimized	79.8±0.03	49.8±0.03	0.027±0.003	0.088±0.007	**0.784±0.011**	2.236±0.021
3* Multi-Pass	Deep Ensembles†	80.2±0.01	**54.1±0.04**	0.021±0.004	0.138±0.013	0.666±0.02	2.281±0.03
	MC Dropout†	79.6±0.02	42.6±0.08	0.050±0.003	0.202±0.010	0.825±0.01	2.881±0.01
	Ours β' Grid-Searched	**83.09**	**54.1**	**0.018**	**0.086**	**0.614**	**2.042**

The ultimate goal of the thesis is to improve model calibration under distribution shift by improving sensitivity. Popular metrics for measuring calibration include: Negative Log-Likelihood (**NLL** (27)), **Brier** (11) and Expected Calibration Error (**ECE** (50)). Our goal is for our model is to produce values close to 0 in these metrics, which maximizes calibration. Please refer to Sec. 3.1 for more detailed discussion on these metrics. Following prior works (43; 65; 52), we will use CIFAR10 and CIFAR100 as the in-distribution training and testing dataset, and apply the image corruption library provided by (29) to benchmark calibration performance under distribution shift. The library provides 16 types of noises with 5 severity scales. In this section, we show that our model outperforms other deterministic methods (despite their significant complexity

Table 5.4: **Generalizability Experiments** Our method is effective with different feature backbones.

		Clean				Corrupt/Rotate			
model	dataset	accuracy↑	ECE↓	NLL↓	Brier↓	accuracy↑	ECE↓	NLL↓	Brier↓
ResNet34	CIFAR10	95.63%	0.026	0.186	0.007	**81.96**%	0.164	1.114	0.039
GSD ResNet34	CIFAR10	**95.9**%	**0.005**	**0.148**	**0.006**	76.54%	**0.088**	**0.882**	**0.037**
ResNet50	CIFAR10	95.32%	0.03	0.203	0.008	**76.32**%	0.17	1.23	0.039
GSD ResNet50	CIFAR10	**95.82**%	**0.008**	**0.147**	**0.007**	76.23%	**0.057**	**0.766**	**0.033**
ResNet101	CIFAR10	95.61%	0.028	0.197	**0.007**	77.59%	0.154	1.118	0.037
GSD ResNet101	CIFAR10	**95.62**%	**0.007**	**0.158**	**0.007**	**77.21**%	**0.075**	**0.852**	**0.036**
ResNet152	CIFAR10	**95.7**%	0.028	0.196	**0.007**	75.2%	0.179	1.337	0.041
GSD ResNet152	CIFAR10	95.63%	**0.007**	**0.151**	**0.007**	**76.58%**	**0.058**	**0.765**	**0.033**
ResNet34	CIFAR100	**78.81**%	0.071	**0.868**	**0.003**	**51.16**%	0.19	2.387	**0.007**
GSD ResNet34	CIFAR100	78.02%	**0.037**	0.938	**0.003**	49.27%	**0.098**	**2.361**	**0.007**
ResNet50	CIFAR100	**79.28**%	0.075	**0.861**	**0.003**	49.71%	0.213	2.477	0.007
GSD ResNet50	CIFAR100	78.97%	**0.033**	0.879	**0.003**	**50.12**%	**0.08**	**2.264**	**0.006**
ResNet101	CIFAR100	**80.17**%	0.092	0.846	0.003	**58.19**%	0.253	2.575	0.007
GSD ResNet101	CIFAR100	79.82%	**0.034**	**0.834**	**0.003**	53.14%	**0.082**	**2.11**	**0.006**
ResNet152	CIFAR100	**80.71**%	0.090	**0.815**	**0.003**	**54.2**%	0.233	2.45	0.007
GSD ResNet152	CIFAR100	79.85%	**0.036**	0.827	**0.003**	53%	**0.078**	**2.12**	**0.006**

Table 5.5: **Importance of Norm** While norm is poorly calibrated, it is important for calibration.

	ECE	NLL	Brier	Entropy	Accuracy
Vanilla ($\|\mathbf{w}_y\|\|\mathbf{x}\| \cos \phi_y$)	0.025±0.001	0.186±0.006	0.001±0.0	0.082±0.002	95.4±0.1%
No Weight Norm (w/o $\|\mathbf{w}_y\|$)	0.061±0.0003	0.206±0.006	0.001±0.0	0.527±0.014	95.4±0.1%
No x Norm (w/o $\|\mathbf{x}\|$)	0.893±0.002	2.837±0.005	0.009±0.0	4.537±0.001	95.4±0.1%
Only Cosine (w/o $\|\mathbf{w}_y\|$,$\|\mathbf{x}\|$)	0.914±0.001	3.235±0.001	0.009±0.0	4.546±0.000	95.3±0.1%

Compared Methods We compare to several popular state-of-the-art models including stochastic Bayesian methods (multi-pass): Deep Ensemble (37) and MC dropout (18), and recent deterministic methods (single pass): SNGP (43) and DUQ (65).

Results In Tab. 5.2 and 5.3, we compare our model to the most recent state of art deterministic methods SNGP and DUQ using Wide ResNet 28-10 (70) as the model backbone and each model evaluated using the average of 10 seeds. We report accuracy, ECE and NLL on clean and corrupted CIFAR10/100 datasets (29). Our method outperforms all single-pass methods on calibration when data is corrupted, and even surpass ensembles on error metrics for corrupted data. We had 2 versions of our model: **Grid Searched:** grid search β' on the validation set to minimize ECE and **Optimized:** optimize β' on the validation set via gradient decent to minimize NLL for 10 epochs, similar to temperature scaling.

Generalizability We explored how generalizable our method (Grid Searched) is by applying it to 12 different models and 4 different datasets in Tab. 5.4. We can see consistently that our model had stronger calibration across all models and metrics, including models known to be well calibrated like LeNet (39). All models were tested on CIFAR10C and CIFAR100C datasets offered by (29) where the original CIFAR10 and CIFAR100 were pre-corrupted; these were used for consistent corruption benchmarking across all models. All non-CIFAR datasets were corrupted via rotation from angles [0,350] with 10 step angles in between and the average calibration and accuracy were taken across all degrees of rotation. Our models included: DenseNet (30), LeNet (39), and 6 varying sizes of ResNet, which are described in (28). The datasets we experimented on CIFAR10 (36), CIFAR100 (36), MNIST (40) and SVHN (51), CIFAR10C (29), CIFAR100C (29).

Qualitative Comparison The current state-of-the-art single pass models for inference on OOD data, without training on OOD data, are SNGP (43) and DUQ (65). The primary disadvantages of these models are: **1) Hyperparameter Combinatorics:** Both DUQ and SNGP require many hyperparameters. Our model only has *one* hyperparameter that is tuned post-training, which is quicker and less costly than the other methods that require pre-training tuning. **2) Extended Training Time:** DUQ requires a centroid embedding update every epoch, while SNGP requires sampling potentially high dimensional embeddings of training points, thus increasing training time while our model

Table 5.6: **Pearson Correlation of Cosine Similarity and Norm vs. ECE during training on CIFAR100**. Norm is consistently positively correlated with ECE whereas the similarity is either negatively or not correlated with ECE.

	ResNet18			ResNet34			ResNet101			ResNet152		
	shot	Gaussian	Defocus	shot	Gaussian	Defocus	shot	Gaussian	Defocus	shot	Gaussian	Defocus
Cosine Sim	0.09	0.03	0.73	0.09	0.03	0.32	-0.03	-0.04	-0.88	-0.97	0.04	-0.81
Norm	**0.82**	**0.82**	**0.78**	**0.82**	**0.81**	**0.78**	**0.87**	**0.87**	**0.85**	**0.86**	**0.85**	**0.81**

trains in the same amount of time as the model it is applied to. Bayesian MCDO (18) and Deep Ensemble (37) are considered the current state-of-the-art methods for multi-pass calibration. Bayesian MCDO requires multiple passes with dropout during inference. Deep Ensembles requires N times the number of parameters as the single model it is ensembling where N is the number of models ensembled. The main disadvantage of multi-pass models is high inference complexity while our model adds no overhead computation at inference.

Importance of the Norm While we have shown and conjectured that the norm of x is uncalibrated to OOD data and not always well calibrated to IND data, one might suggest simply removing the norm. We show in Tab. 5.5 though the norm is uncalibrated it is still important for inference. We trained ResNet18 on CIFAR10 and then ran inference with ResNet18 modified in the following: dividing out the norms of the weights for each class, dividing out the norm of the input, and then dividing out both. As we can see the weight norm contributes minimally to inference as accuracy decreased by 0.03% without it and as previous work has shown the angle dominates classification. We can see with $||\mathbf{x}||$ removed the entropy is at its highest while calibration is very poor, implying the distribution is much more uniform when it should be peaked, as a larger entropy implies a more uniform distribution. Thus the root of the issue does not lie in the existence of the norm, but in's lack of sensitivity.

5.2.2 Reasons for Bad Calibration under Distribution Shift

To identify the cause of bad calibration, we record the accuracy, ECE, norm, and cosine similarity of a model during the training of a vanilla ResNet model. Specifically, we record the evaluation statistics on clean data and also on data corrupted with Gaussian noise on CIFAR100. Fig. 5.2a and 5.2b show the accuracy and ECE respectively. We observe that evaluation on Gaussian noise corrupted data yields lower accuracy and higher ECE compared to evaluation on clean data. *This demonstrates that the model's confidence*

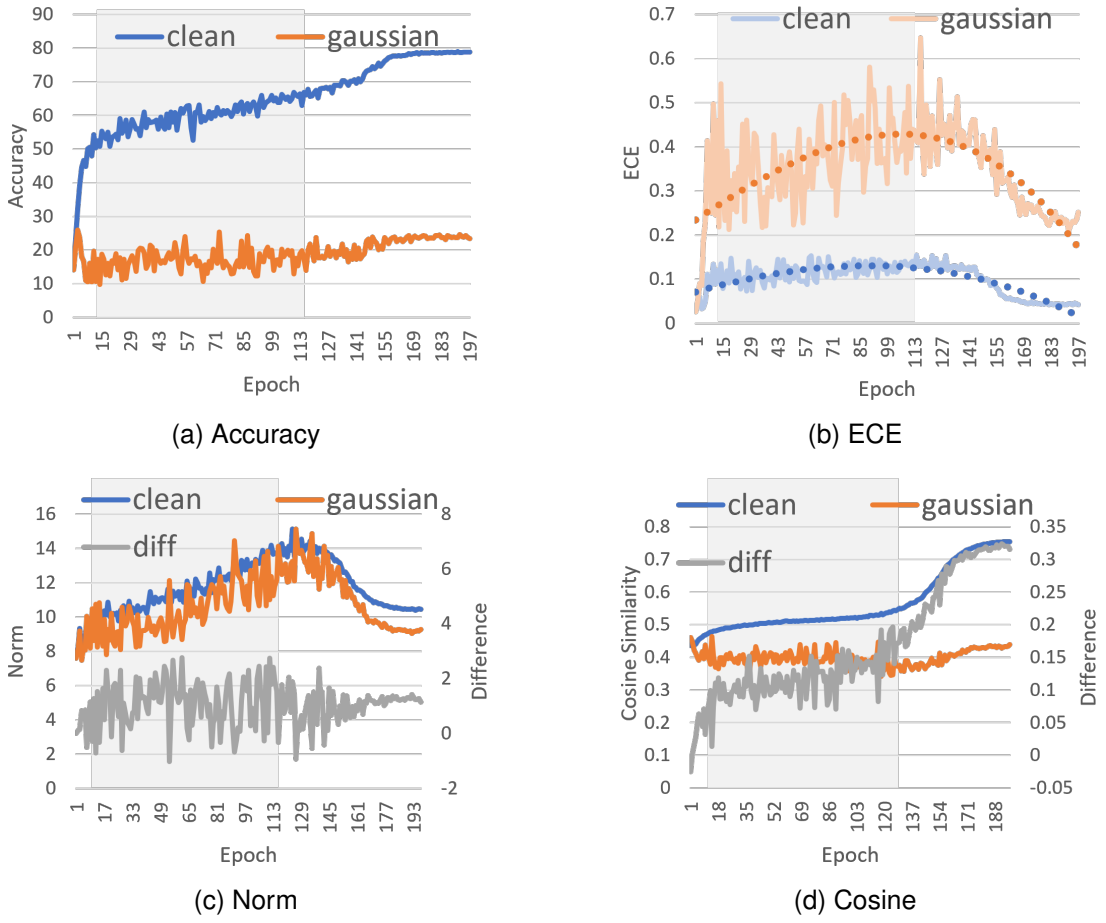


Figure 5.2: **Accuracy, ECE, norm, and cosine similarity on CIFAR100 validation set with clean and Gaussian noise trained on vanilla ResNet.** In the shaded region, an increase in norm is responsible for the increase in ECE because cosine similarity is relatively flat. Throughout training, sensitivity of the cosine similarity improves while that of the norm remains insensitive.

fails to adapt to the decreasing accuracy. Fig. 5.2c and 5.2d show the change of average norm and average cosine similarity throughout training. The difference between Gaussian noised data and clean data is also reported. We observe that the norm of clean data and the norm of Gaussian noised data are close and the difference remains constantly low whereas the cosine similarity of the two diverges with training. *This indicates that the sensitivity of cosine similarity increases whereas the sensitivity of the norm remains low with training.* In the shaded region of Fig. 5.2b-5.2d where ECE increases the most, we observe that the norm also increases but the cosine similarity only increases slowly. The observation also holds for other noises and architectures. We further present Pearson correlation between ECE and cosine similarity or norm on 4 models and 3 noises in Tab. 5.6. A large correlation coefficient indicates a higher positive correlation. Norm is consistently positively correlated with ECE whereas the similarity is either negatively or not correlated with ECE. This shows that the worsening of ECE (large ECE) is correlated with the increasing norm. Based on supporting literature (33), (13) and this correlation, the observation supports the conjecture that the insensitivity of the norm is responsible for bad calibration.

5.2.3 Empirical Support for the Disentangled Training

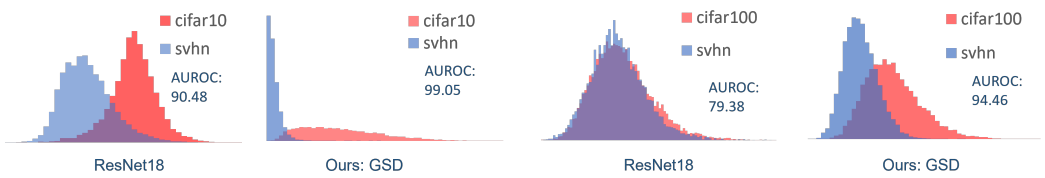
Table 5.7: **OOD AUROC↑ using Norm and Similarity** We show OOD detection results using norm and cosine similarity. SVHN (51) is used as the OOD dataset. Our method (α-regularized) significantly increases the sensitivity of feature norm.

ResNet18	Criterion	CIFAR10	CIFAR10 (Incorrect)
2*Vanilla	Norm	90.48	67.23
	Similarity	93.87	56.98
2*α- regularized	Norm	**99.05**	**93.16**
	Similarity	97.09	74.82
2*α- unregularized	Norm	98.20	88.29
	Similarity	94.72	60.63

(a) **CIFAR10 vs. SVHN AUROC**

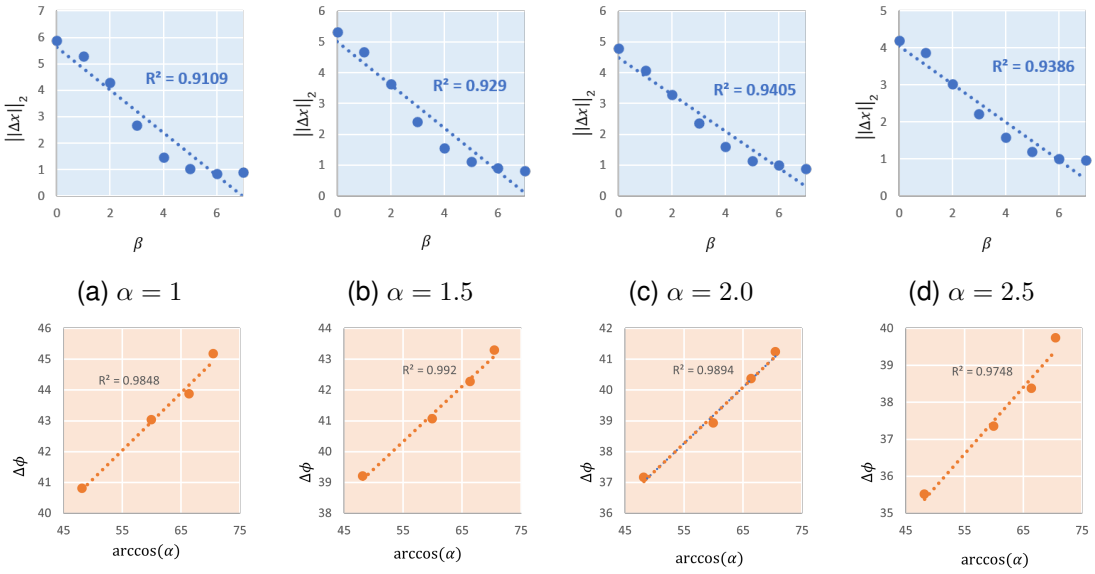
ResNet18	Criterion	CIFAR100	CIFAR100 (Incorrect)
2*vanilla	Norm	79.38	62.66
	Similarity	82.26	55.54
2* α- regularized	Norm	**94.46**	**86.67**
	Similarity	85.68	63.24
2*α- unregularized	Norm	84.78	73.11
	Similarity	72.61	42.90

(b) **CIFAR100 vs. SVHN AUROC**



(a) CIFAR10 vs. SVHN (b) CIFAR100 vs. SVHN

Figure 5.3: **Histogram of Norm Distribution** Our model (α-regularized) improves the separation of norm between IND and OOD data.



(a) $\alpha = 1$ (b) $\alpha = 1.5$ (c) $\alpha = 2.0$ (d) $\alpha = 2.5$

(e) $\beta = 0$ (f) $\beta = 2$ (g) $\beta = 2$ (h) $\beta = 3$

Figure 5.4: **Properties of $\|\Delta x\|_2$ and $\Delta\phi$.** (a) - (b): $\|\Delta x\|_2$ decreases linearly with β for fixed α reflecting Eq. 5.2 and 5.6. (e) - (h) $\Delta\phi$ increases linearly with $arccos(\alpha)$ for fixed β reflecting Eq. 5.3 and 5.6. All plots include R-squared values to indicate the goodness-of-fit of the linear relationship.

In the first set of experiments, we show that α and β reflect the effects of the geometric decomposition as claimed in Sec. 5.1.2 with different $\alpha - \beta$ configurations. From Fig. 5.4a - 5.4d, we observe that the norm decreases linearly with β for fixed α. From Fig. 5.4e - 5.4h, we observe that the angle increases linearly with $arccos(\alpha)$. The observations are consistent with the original geometric motivation. β encodes an instance-independent portion, $\mathcal{C}_x$, of the norm. As β increases, $\mathcal{C}_x$ increases and therefore the magnitude of the dependent component, $\|\Delta x\|_2$ decreases linearly. α

Table 5.8: **Average norm and accuracy across different corruptions on GSD ResNet18**. The table is organized in decreasing accuracy order.

ResNet GSD	clean	brightness	fog	elastic	snow	defocus	frost	motion blur	jpeg	zoom blur	pixelate	contrast	shot	glass blur	impulse	Gaussian
accuracy	95.33	93.82	88.75	85.09	83.87	82.95	80.41	79.71	79.31	78.48	76.4	75.29	59.46	59.29	57.26	47.33
norm	0.73	0.66	0.52	0.42	0.46	0.46	0.44	0.37	0.39	0.35	0.5	0.39	0.34	0.27	0.3	0.28

encodes the inverse of the cosine of a relaxation angle, $\mathcal{C}_\phi$. As $arccos(\alpha)$ increases, the resulting angle, $\Delta\phi$ increases linearly due to the increased relaxation angle encoded by α.

In the second set of experiments, we show that the new model effectively increases the sensitivity of both the norm and the angle to input distribution shift as claimed in Sec. 5.1.3. Specifically, we measure OOD detection performance of the models using both the norm and the cosine similarity with the Area Under the Receiver Operating Characteristic (AUROC) curve metric. We use CIFAR10/100 as the IND data and SVHN (51) as the OOD data. In Tab. 5.7a and 5.7b we show two configurations of models in addition to vanilla ResNet18: (α-regularized) we regularize α such that it stays close to one as described in Sec. 5.1.3; (α-unregularzed) we optimize both α and β freely without constraints. Compared to vanilla ResNet, the norms predicted by our models achieve significant improvement in separating IND data from OOD data. Additionally, we visualize the distribution of norms in Fig. 5.3a and 5.3b. The separation between IND and OOD data increases significantly compared to vanilla ResNet18. However, a large α (see α-unregularzed in Tab. 5.7a and 5.7b) leads to marginal cosine similarity sensitivity improvement on CIFAR10 and CIFAR100. This indirectly confirms our observations in Sec. 5.2.2 and prior works (13) that cosine similarity correlates well with distribution shift. Introducing further angle relaxation might not be always beneficial. While we mainly focus on calibration, our method also strengthens its base model's ability for OOD detection.

The assumption that OOD data have smaller norms is based on the expectation that a model should be less confident in OOD data. Practically, the norm acts as a temperature in softmax as shown in Eq. 5.1. Intuitively, larger always yields more peaked/confident predictions, and smaller always yield flatter predictive distributions. Therefore, we expect less confident data such as OOD data to have smaller because we expect the output distribution to be flatter. The assumption is supported by the following empirical evidence. In Tab. 5.8 we show the norm of in-distribution and out-of-distribution data on CIFAR10 using ResNet50-GSD (ours). The OOD data is produced by the 15 corruptions used in (29). OOD data have consistently smaller norms and the accuracy decreases with decreasing norms with a Pearson correlation of 0.9 as an indicator of more out-of-distribution.

5.3 Summary

We studied the geometry of the last linear decision layer and identified the insensitivity of the norm as the culprit of bad calibration under distribution shift. To encourage sensitivity, we derived a general theory to decompose the norm and angular similarity. Inspired by the theory, we proposed a simple yet very effective training and inference scheme that encourages the norm to reflect distribution changes. The model outperforms other deterministic single-pass methods in calibration metrics with much fewer hyperparameters. We also demonstrated its superior generalizability on a variety of popular neural networks. Note that our problem and method have a positive societal impact, as calibration under shift improves the overall confidence and robustness of these models. In the next chapter we examine another inherent issue with CNNs, which is their bias toward high-frequency features (19). The effects of this when CNNs are applied to RL are that the agent doesn't learn task-relevant features.

Chapter 6 Augmentation Curriculum Learning

Reinforcement Learning (RL) has shown great success in a large variety of problems from video-games (48), navigation (66), and manipulation (41; 32) even while operating from high-dimensional pixel inputs. Despite this success, the policies produced by RL are only well suited for the same environment they were trained for and fail to generalize to new environments. Instead, agents overfit to task-irrelevant visual features, resulting in even simple visual distortions degrading policy performance. A key objective of image-based RL is building robust agents that can generalize beyond the training environment. Unlike Ch. 5 where we explored a method that imbued calibration via increasing the sensitivity of the final output layer and regularizing the norm, the methods in this chapter aim to regularize the CNN using image augmentation in order to achieve the current best policy, while leveraging semantically similar augmentations as an approximation to the distribution shift we'd see in deployment.

Several existing approaches to training more robust agents include domain randomization (54; 64) and data augmentation (26; 25; 17). Domain randomization modifies the training environment simulator to create more varied training data, whereas data augmentation deals with augmenting the image observations representing states without modifying the simulator itself. Prior work shows pixel-based augmentation improves sample efficiency and helps agents achieve performance matching state-based RL (38; 68). Therefore, in this work, we focus on data augmented generalization to visual distribution shift while the semantics remain unchanged. We specifically aim to do this in a zero-shot manner, i.e., where shifted data is unavailable during training.

Unlike for supervised and self-supervised image classification tasks, augmentation for pixel-based RL has demonstrated mixed levels of success. Prior work categorized augmentations into weak and strong ones based on downstream training performance (17). Specifically, works define **weak augmentations** as those allowing the agent to learn a policy with higher episodic rewards in the training environment than

training without augmentation. **Strong augmentations** refers to augmentations that lead to empirically worse performance than training with no augmentations. Classifying augmentations according to this definition is dependent on the task. For example, cut-out color has empirically been shown to be detrimental ("strong augmentation") for all tasks in Deep Mind Control Suite (DMC) (63), but is a effective ("weak augmentation") for Star Pilot in Procgen (14) as shown in (38). Methods exist that attempt to automate finding the optimal weak augmentation on a per-task basis (56), but these still do not expand the effectiveness of many augmentations.

Many RL generalization methods leverage weak augmentation for better policy learning training and add strong augmentations in training for generalization to visual distribution shift (26; 25; 17). However, these methods suffer from strong augmentation making training harder due to the difficulty of learning from such diverse visual observations, destabilizing training. This results in strong augmentations causing the agent to not learn a policy with as strong performance as using weak augmentations alone. In this work, we introduce a new training method that avoids the training instabilities caused by strong augmentations through a curriculum that separates augmented training into weak and strong training phases. Once the network has been sufficiently regularized in the weak augmentation phase, it is cloned to create a policy network that is trained on strong augmentations. This disentangles the responsibilities of the networks into accurately approximation the Q-value (network trained on weak augmentations) of the agent and generalization (doing well on shifted test distributions). Crucially we separate the two networks to avoid the destabilizing effect of strong augmentations. We also demonstrate the power of the method under even more severe augmentation, namely a new splicing augmentation that pastes relevant visual features into an irrelevant background. We show that our curriculum learning approach can effectively leverage strong augmentations, and the combination of our method with this new augmentation technique achieves state-of-the-art generalization performance.

Our main contributions are summarized as follows:

- We introduce Augmentation Curriculum Learning (AugCL), a new method for learning with strong visual augmentations for generalization to unseen environments in pixel-based RL.

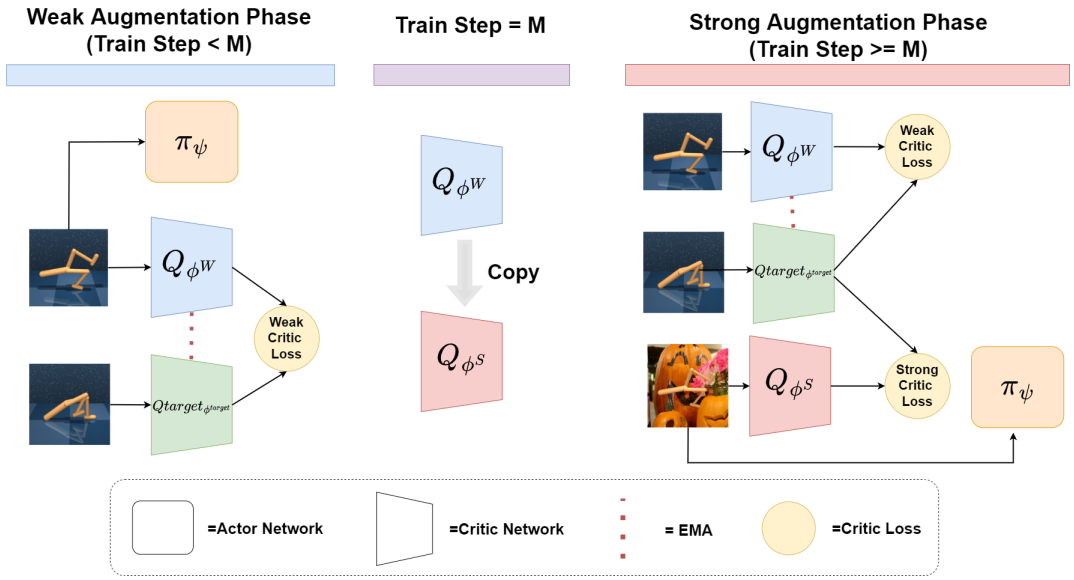


Figure 6.1: Neural architecture and tensor flow across different phases for AugCL.

- A new visual augmentation named Splice, which by simulating distracting backgrounds helps prevent overfitting to task irrelevant features.
- We demonstrate AugCL achieves state-of-the-art results across a suite of pixel-based RL generalization benchmarks.

This work was done in collaboration with Andrew Szot, Prithvijit Chattopadhyay and will be submitted to ICML 2023.

6.1 Method

6.1.1 Agent Architecture

The key differences between AugCL and previous works are **1)** We train a weak and strong augmented network in parallel. **2)** We train only on weak augmentation for the early phases of training. **3)** We bootstrap the strong augmented network from a weakly augmented Q target network. The driving intuition behind this is strong augmentations incur non-zero degradation to policy learning but help with generalization. Hence to mitigate this issue, we have a separate network trained only on strong augmentation and update the target network using EMA from the weak augmented network. This duo circumvents the policy degradation by allowing the weak critic and Q target network to learn a state and action value approximation without being impeded by strong augmentation. Then by bootstrapping this Q target network with an accurate approximation of

the value function, the strong augmented network learns to generalize under strong augmentation. The weak augmented pre-training is required to regularize the CNN from biasing itself to high-frequency features. We show both are necessary to achieve SOTA performance in 6.2.3.

AugCL builds off of SAC, learning a critic and policy network as shown in Fig. 6.1. The critic network $Q_\phi(s_t, a_t)$ takes as input a stacked sequence of image observations, and an action produces the expected return for taking action a_t in state s_t. We also learn a parameterized policy $\pi_\psi(s_t)$ that outputs a normal distribution parameterized by a learned mean and variance which then samples an action for the current state. Q_ϕ and π_ψ share a visual encoder $h_\mu(s_t)$ which takes as input the high-dimensional image observations and produces a low-dimensional state encoding. The parameters of ψ and ϕ both include the encoder parameters μ, which are updated as a part of both the policy and critic losses.

AugCL trains Q_ϕ and π_ψ through the standard SAC losses with random image augmentations. We sample augmentations f from a distribution over augmentations $\mathcal{F}$. We then use f to augment the states s_t in the SAC losses from Sec. 2.16. We augment only the current state s_t in the SAC loss similar to (25). The resulting losses for the critic and policy with the augmentations are respectively:

$$\mathcal{L}_Q(\phi; \phi^{target}, \mathcal{F}) =_{\tau\sim D, f\sim\mathcal{F}} \left[(Q_\phi(f(s_t), a_t) - (r_t + \gamma V(s_{t+1}; \phi^{target})))^2\right] \tag{6.1}$$

$$\mathcal{L}_\pi(\psi; \phi, \mathcal{F}) = -_{a\sim\pi, f\sim\mathcal{F}}[Q_\phi(f(s_t), a) - \alpha \log \pi_\psi(a|f(s_t))] \tag{6.2}$$

Notice that the losses are defined relative to a distribution over augmentations $\mathcal{F}$. The choice of this augmentation distribution is an important consideration in the algorithm's stability and robustness to new MDPs. Prior work breaks up augmentations for pixel-based control into two classes: weak augmentations and strong augmentations.

Weak Augmentations (denoted as $\mathcal{F}^W$) are augmentations that help stabilize and improve training performance in the source MDP $\mathcal{M}$, but are insufficient for generalization to new MDPs $\overline{\mathcal{M}}$. (12) shows that training with weak augmentations is important to prevent overfitting to high-frequency features in the image space when learning from bootstrapped targets in actor-critic methods. The consequences of this overfitting have been shown to cause the critic network to overfit to its own predictions, and a decrease in

correlation to Monte Carlo returns as training ensues. Training with weak augmentations is therefore an important part of any actor-critic control-from-pixels method, such as AugCL. Known weak augmentations for all DMC tasks are: crop, translate (38) and shift (68).

Strong Augmentations (denoted as $\mathcal{F}^S$) are augmentations that help improve policy performance in new MDPs $\overline{\mathcal{M}}$. Visual perturbations such as random color changes of the environment can be simulated with strong augmentations such as: random convolution and random color jitter. While distracting backgrounds can be simulated with mix-up (71), these augmentations are difficult to train with, as they increase the difficulty of the learning problem due to the duo of stochastic parameter sampling for $\mathcal{F}^S$ and high visual variance between samples. AugCL addresses how to effectively incorporate strong augmentations such as: random convolution, overlay (variation of mix-up), and our novel augmentation into training.

6.1.2 AugCL: Curriculum Learning with Strong Augmentations

As mentioned, strong augmentations are detrimental to learning but important to train with for generalization performance. The key idea of our method is therefore to leverage curriculum learning (6) to avoid this destabilization. Specifically, AugCL defines a curriculum over augmentations to enable better training and generalization. It is well known that CNNs are inherently biased to high-frequency features (19; 31), the consequences of this in RL is a lower average episodic reward in the train environment. AugCL uses weak augmentations early in training to regularize the CNN. Then later in training AugCL introduces strong augmentations to improve the robustness of the policy. We train two separate networks in parallel as we believe that strong augmentation incurs a non-zero degradation to policy learning, as shown in 6.2.3.

AugCL is described in Alg. 1. AugCL begins by acting in the environment with π_ψ and then adding the observed transition to the replay buffer (lines 5-7). We then sample data batches from the replay buffer for updating the policy and critic. Our curriculum learning schedule breaks the updates into two phases. For the first M policy updates, AugCL is in the *weak augmentation phase* and updates the critic and policy from weak augmentations alone (lines 11,16). Target critic parameters ϕ^{target} are updated as exponential moving averages of the learned critic parameters ϕ (line 17), and used in the bootstrap term of the critic loss. The purpose of the weak augmentation phase

Algorithm 1 AugCL

1: $\phi^W, \psi, \phi^{target}$: critic parameters, policy parameters and Q-target parameters
2: α, β, ζ: actor learning rate, critic learning weight and momentum encoder weight
3: M: Update step to switch to strong augmentation
4: **for** timestep $t = 1, ..., T$ **do**
5: $\quad a_t \sim \pi_\psi(\cdot|s_t)$ ▷ Sample action
6: $\quad s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ ▷ Step environment
7: $\quad \mathcal{B} \leftarrow \mathcal{B} \bigcup (s_t, a_t, r(s_t, a_t), s_{t+1})$ ▷ Add transition to replay buffer
8: $\quad$ **if** $t = M$ **then**
9: $\quad\quad \phi^S = \phi^W$ ▷ Clone critic
10: $\quad$ **end if**
11: $\quad \{s_i, a_i, r_i, s_{i+1} | i = 1, ..., N\} \sim \mathcal{B}$ ▷ Sample transition
12: $\quad \phi^W \leftarrow \phi^W - \beta \nabla_\phi \mathcal{L}_Q(\phi^W; \mathcal{F}^W, \phi^{target})$ ▷ Weak Critic loss
13: $\quad$ **if** $t \geq M$ **then**
14: $\quad\quad \psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}_\pi(\psi; \mathcal{F}^S)$ ▷ Strong Actor loss
15: $\quad\quad \phi^S \leftarrow \phi^S - \beta \nabla_\phi \mathcal{L}_Q(\phi^S; \mathcal{F}^S, \phi^{target})$ ▷ Strong Critic loss
16: $\quad$ **else**
17: $\quad\quad \psi \leftarrow \psi - \alpha \Delta_\psi \mathcal{L}_\pi(\psi; \mathcal{F}^W)$ ▷ Weak Actor loss
18: $\quad$ **end if**
19: $\quad \phi^{target} \leftarrow (1 - \zeta)\phi^{target} + \zeta \phi^W$ ▷ Q-target EMA update
20: **end for**

is to stabilize policy learning. Prior work shows that it is easy for the critic to overfit in image-based RL and weak augmentations are important to achieve strong training performance (12). However, the weak augmentations do not make the policy robust to new visuals. Improving generalization performance is the purpose of the next *strong augmentation phase*.

Then, after M policy updates, AugCL switches to the *strong augmentation phase* and incorporates strong augmentations into training (lines 12-14). A new strong critic network ϕ^S is copied from the weak critic network ϕ^W (line 9). Now, the weak critic network is updated like in the previous phase by training with weak augmentations. However, the separate strong augmentation network with parameters ϕ^S is now trained with strong augmentations (line 14). The policy is also updated with the strong augmentations (line 1) . Separating the strong and weak augmentations into two networks is important for stability. The weak critic helps stabilize bootstrap targets by leveraging weak augmentation to better approximate the state, action value function while the strong critic focuses on generalization performance. Previous methods have attempted this parallel training of the weak and strong augmented network, but with little success (17). We show that this is due to not regularizing the CNN encoder first in Sec. subsec:ablations. A figure of the architecture and the flow of tensors representing input and output of each neural

layer can be seen in Fig. 6.1.

The advantage of AugCL separating strong augmentations into a later phase of learning is it does not require a delicate balance between potentially conflicting losses from strong and weak augmentations. SODA and SVEA incorporate strong augmentations as an auxiliary learning signal that is always applied in conjunction with learning an accurate approximation of the state, action value from weakly augmented data. By learning from both data at the same time, the networks must contend with the trade off between stronger augmentations improving generalization yet harming training performance. The auxiliary objective in SODA may suffer from gradient interference from the conflicting losses as the critic network is optimized to learn an accurate state, action value approximation, and a contrastive loss in parallel. SVEA suffers from a similar issue in that it requires a hyperparameter to balance the combination of losses from strong and weak augmented data. On the other hand, AugCL disentangles the responsibilities of state, action value approximation, and generalization between the weak augmented critic and the strong augmented critic respectively. We empirically demonstrate this by showing AugCL performs better than SVEA and SODA on a variety of benchmarks and has minimal train environment performance degradation, as shown in 6.2.4. Also note SODA and SVEA only pass weakly augmented observations to the policy network during training, the issue with this is that $\pi_\phi(\mathcal{F}^W(s_t)) \neq \pi_\phi(\mathcal{F}^S(s_t))$. AugCL does not suffer from this issue as we pass $\mathcal{F}^S(s_t)$ through the policy network at train time, and interestingly we find train performance still improves and converges to a marginally worse performance on the train environment than training with weak augmentation alone as shown in 6.2.4.

6.1.3 Splice Augmentation

Since AugCL is well suited to train with challenging strong augmentations, we introduce a novel augmentation called "**Splice**" to improve generalization in visual RL. RL generalization benchmarks that incorporate background distractions (26; 61) are challenging for state-of-the-art visual RL approaches. The standard solution is to introduce a variation of mix-up augmentation (71) to RL training (26; 17; 25). (26) theorized that previous strategies failed to adapt to severe background distractions because task-relevant visual features such as the agent's shadow were removed.

Our new augmentation **Splice** solves this issue by pasting relevant visual features

into an irrelevant background. This explicit separation of task-relevant versus task-irrelevant features helps generalization. Specifically, we mask out all non-relevant parts of the visual observation through a segmentation mask which is available in the simulation. We then replace all the non-task parts of the image with a random background image. We use COCO (42) for our experiments as the background replacement images.

6.1.4 More Details on Splice Augmentation

The inspiration for Splice came when we noticed that in many robotics task the relevant visual features had higher brightness. We noticed that DMC fit this criteria well as the ground and background tended to be a dark blue, while the agent is a combination of bright colors (typically yellow and a bright blue). Splice converts an RGB image to HSV color space then sets a threshold for hue, saturation and value. We use kornia [(57)] for color space conversion. Hue represents color, saturation represents chromatic intensity and value represents brightness. If all values in a cell in the HSV converted image exceed the preset thresholds then they are imparted on a new image. In our case we splice out the agent and paste it onto a randomized background. (38; 12) show weak augmentation leads to a better spatial attention mapping of features the agent can control like the robot over high frequency features like the background and flooring. Splice has the ability to impart human prior knowledge about the tasks through tuning the thresholds. By tuning the thresholds accordingly the user can parse out only the relevant visual features in the task. This allows us to circumvent the high frequency feature bias that CNNs inherently have by pasting the relevant features on random backgrounds, thus the high frequency features between frames becomes the task relevant features. Example code is given below an a comparative example is shown in Fig. 6.2.

```
import torch
import kornia


def splice(x, hue_t, saturation_t, value_t):
    b, _, h, w = x.shape
    x_HSV = kornia.color.rgb_to_hsv(x)
    overlay = sample_background(batch_size=b)
    thresholds = torch.FloatTensor([hue_t, saturation_t, value_t])
```

(a) Video Hard

(b) Splice

Figure 6.2: Value threshold of 0.6, with threshold of 0 set for hue and saturation for splice augmentation shown in 6.2b. 6.2a taken from DMC-GB.

```
thresholds = thresholds.view(1, -1, 1, 1).repeat(b, 1, h, w)
mask = x_HSV > thresholds
mask = torch.all(mask, dim=1)
overlay[mask] = x[mask]
return overlay
```

6.2 Experiments

We now evaluate AugCL and baselines on how well they can generalize to visual distribution shifts in the DMControl Generalization Benchmark (DMC-GB). In Sec. 6.2.1, we describe the experimental setup for how our method and baselines are configured. Next, in Sec. 6.2.2, we show that AugCL achieves state-of-the-art performance in the majority of settings in DMC-GB. Finally, in Sec. 6.2.3, we analyze what hyperparameters are necessary for the benefits of AugCL.

6.2.1 Experimental Setup

Environments and Evaluation: The purpose of our experiments is to evaluate how well policies trained with various methods can generalize to new visual disturbances. All methods are first trained in a source environment without any visual disturbances. We then evaluate the trained policy in the same environment but with random visual disturbances. DMC-GB tests how methods can generalize to random colors, backgrounds, and camera poses. All methods are trained for 500,000 frames and evaluated on 5

Table 6.1: Results from DMC-GB benchmark color hard. All methods are evaluated on 5 seeds over 30 episodes. The mean and standard deviation are provided. AugCL outperforms baseline in 4 out the 5 tasks.

Domain, Task	CURL	RAD	DrQ	PAD	SODA (conv)	SVEA (conv)	AugCL (conv)
Walker, Walk	445 ± 99	400 ± 61	520 ± 91	468 ± 47	697 ± 66	760 ± 145	$\mathbf{890 \pm 36}$
Walker, Stand	662 ± 54	644 ± 88	770 ± 71	797 ± 46	930 ± 12	942 ± 26	$\mathbf{956 \pm 17}$
Cartpole, Swingup	454 ± 110	590 ± 53	586 ± 52	630 ± 63	831 ± 21	837 ± 23	$\mathbf{852 \pm 9}$
Ball In Cup, Catch	231 ± 92	541 ± 29	365 ± 210	563 ± 50	892 ± 37	$\mathbf{961 \pm 7}$	957 ± 18
Finger, Spin	691 ± 12	667 ± 154	776 ± 134	803 ± 72	901 ± 51	977 ± 5	$\mathbf{980 \pm 9}$

Table 6.2: Hyperparameters used for all experiments

Hyperparameter	Value
Frame Rendering	$3 \times 84 \times 84$
Frames Stacked	3
Random Shift	4 pixels
M	200,000
Action Repeat	2 (finger), 8 (cartpole), 4 (otherwise)
Discount Factor γ	0.99
Episode Length	1000
Learning Algorithm	SAC
Number Of Frames	500,000
Replay Buffer Size	500,000
Optimizer (β)	Adam(β_1=0.9, β_2=0.999)
Optimizer (α)	Adam(β_1=0.5, β_2=0.999)
Learning Rate (θ)	1e-3
Learning Rate (α of SAC)	1e-4
Batch Size	128
$\hat{\phi}$ Update Frequency	2
$\hat{\phi}$ Momentum Coefficient	0.05(encoder), 0.01(critic)
Seeds	[0,4]

tasks from DMC-GB in three different evaluation settings from DMC-GB (color-hard, video-easy, and video-hard). The 5 tasks from DMC-GB used in this paper are described in Tab. 4.1. We report the mean and standard deviation across 5 seeds per method, where each seed is evaluated by taking the average episode return across 30 episodes. For Tab. 6.3 and 6.4 we added training with SVEA during the weak augmentation phase.

Baselines: We compare AugCL against other recent pixel-based RL methods, some of which were explicitly designed for learning robust policies that can generalize to unseen environments. Specifically, we compare against **CURL**, **RAD**, **SVEA**, **SODA**, **DrQ** as well as **PAD** (24), which adapts to the test environment using self-supervision.

Hyperparameters and baseline results are taken from (25). We don't compare to SECANT as it requires double the training frames to all other baselines and requires training 2 models sequentially. Also note that SVEA and SODA augment each batch twice, thus doubling the data the agent trains on whereas AugCL only uses a single batch.

Data Augmentation Setup: We apply random shift (68) as our weak augmentation for AugCL. For all experiments, we selected $M = 200,000$ for AugCL, meaning we first perform 200k updates in the weak augmentation phase before switching to the strong augmentation phase. All hyperparameters shared between AugCL and baselines are kept the same. Random convolution produced the best results in prior works on color hard and overlay for video DMC-GB benchmarks (25), and we therefore use those augmentations for their respective benchmarks. Note that DrQ, AugCL and SVEA all use shift as their weak augmentation and CURL, PAD, RAD and SODA use crop. This is important to note as shift has been shown to give stronger empirical results than crop in DMC tasks (68). "Overlay" in Tab. 6.3, 6.4 refers to (26) version of mix-up. The original SVEA and SODA paper use the Places dataset (72) for Overlay, but during the time this paper was written Places was unavailable due to maintenance, so instead, we used COCO for AugCL. We felt this was a fair comparison as long as both datasets were different from RealEstate10k (73), which is used by DMC-GB. A full list of hyperparameters can be found in Tab. 6.2. We apply random shift (68) as our weak augmentation for AugCL and set $M = 200,000$, which we determined empirically. All overlapping hyper-parameters between methods are kept the same.

We also include results using Splice on the DMC-GB video easy and video hard benchmarks. In the DMControl tasks, we segment out the agent by filtering, converting the RGB image to HSV, and then taking pixels with HSV values only greater than a threshold. We set a consistent value threshold of 0.6 for all tasks to remove all aspects of the image, including the shadow, leaving only the agent. The hue threshold was 0 for all tasks except "Cartpole, Swingup" which required the hue threshold to be set to 3.5 due to the background being a mix of lighter and darker blues in these environments. The saturation threshold was set to 0 for all tasks. The full list of hyperparameters can be found in Tab. 6.2.

Table 6.3: Results from DMC-GB benchmark video easy generalization benchmark.

Domain, Task	CURL	RAD	DrQ	PAD	SODA (splice)	SVEA (splice)	AugCL (splice)	AugCL + SVEA (splice)
Walker, Walk	556 ± 133	600 ± 63	682 ± 89	717 ± 79	625 ± 29	$882 \pm$**63**	$879 \pm$**35**	**904** $\pm$ **29**
Walker, Stand	852 ± 75	745 ± 146	873 ± 83	935 ± 20	955 ± 13	$969 \pm$**4**	958 ± 7	**972** $\pm$ **6**
Cartpole, Swingup	404 ± 67	373 ± 72	485 ± 105	521 ± 76	764 ± 49	850 ± 32	**840** $\pm$**27**	**854** $\pm$ **9**
Ball In Cup, Catch	316 ± 119	481 ± 26	318 ± 157	436 ± 55	$907 \pm$**30**	$963 \pm$ **11**	959 ± 8	**967** $\pm$ **3**
Finger, Spin	502 ± 19	400 ± 64	533 ± 119	691 ± 80	888 ± 160	975 ± 20	**983** $\pm$ **5**	975 ± 17

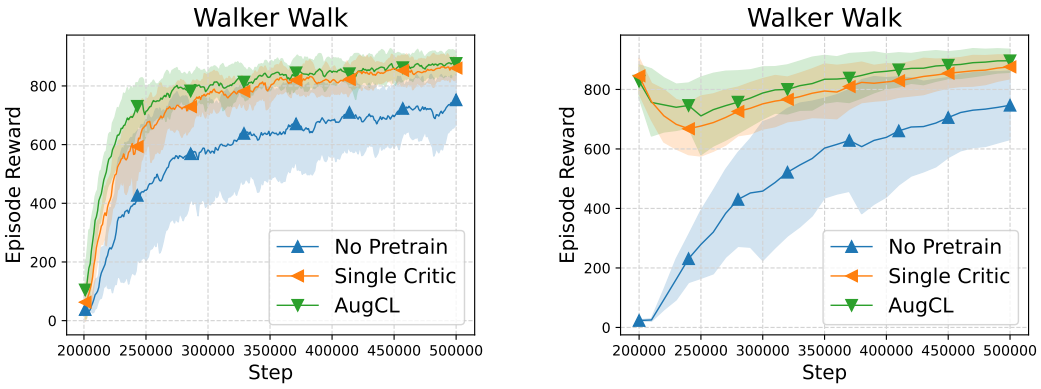
Table 6.4: Results from DMC-GB benchmark video hard.

Domain, Task	CURL	RAD	DrQ	PAD	SODA (splice)	SVEA (splice)	AugCL (splice)	AugCL + SVEA (splice)
Walker, Walk	58 ± 18	56 ± 9	104 ± 22	93 ± 29	619 ± 25	861 ± 59	864 ± 34	**888** $\pm$ **30**
Walker, Stand	45 ± 5	231 ± 39	289 ± 49	278 ± 72	872 ± 71	960 ± 6	959 ± 5	**962** $\pm$ **7**
Cartpole, Swingup	114 ± 15	110 ± 16	138 ± 9	123 ± 24	429 ± 64	776 ± 28	742 ± 35	**784** $\pm$ **16**
Ball In Cup, Catch	115 ± 33	97 ± 29	92 ± 23	66 ± 61	327 ± 100	895 ± 21	**916** $\pm$ **21**	905 ± 35
Finger, Spin	27 ± 21	32 ± 11	71 ± 45	56 ± 18	873 ± 163	948 ± 20	952 ± 24	**960** $\pm$ **17**

6.2.2 DMC-GB Results

Firstly, Tab. 6.1 shows that AugCL outperforms all baselines in **4 out of 5** tasks in DMC-GB color-hard environments. This further closes the gap between the performance of the policy from training and its generalization performance on the test environment. Ball In Cup, Catch and Finger, Spin under color hard are close to matching the current SOTA in the train environment thanks to SVEA and AugCL as shown in Tab. 6.6. Despite SODA and SVEA using the same augmentations as AugCL and also being designed for generalization in pixel-based RL, AugCL outperforms them in evaluation return.

Next, in the DMC-GB video easy and video hard environments AugCL again outperforms baselines in almost all of the settings. AugCL outperforms baselines in **4 out of 5** tasks in video-easy (Tab. 6.3) and in **5 out of 5** tasks in video hard (Tab. 6.4). A combination of the splice augmentation and AugCL performs best. Splice combined with AugCL does well on “Finger, Spin” under video easy as it’s only 1 average episodic



(a) Train

(b) DMC-GB color hard

Figure 6.3: **No Pretrain**: No weak augmented pre-training on the strong network. **Single Critic**: A single critic is used for training under weak and strong augmentation. The line represents the mean over 3 seeds, and the shadow represents variance. Fig. 6.3a shows performance on the train environment, and Fig. 6.3b shows performance during training on DMC-GB color hard. The lines represent averages and the shaded regions the standard deviation of the results across 5 seeds.

reward off from the train environment SOTA as seen in Tab. 6.6. We theorize that Splice performs better than Overlay because Overlay is a weighted sum of pixels from the state image and an irrelevant image. (38; 12) showed the utility of weak augmentation was that a regularized CNN improved spatial attention mapping to task relevant features. Overlay may impede this process by making task relevant features less visible.

6.2.3 Ablations

We analyze two design choices of AugCL: the curriculum and using separate critic networks for weak and strong augmentations. We use Random Convolution as the strong augmentation for all variations of AugCL in this section. **No Pretrain** in Fig. 6.3 represents AugCL, but without the copying of weights to the strongly augmented network at step M (omitting line 9 in Alg. 1). **Single Critic**: represents having a single critic network for both strong and weak augmentation (omitting line 11 and line 17 becomes $\phi^{target} \leftarrow (1-\zeta)\phi^{target} + \zeta\phi^{S}$ in Alg. 1).

As we can see without the pre-training even while bootstrapping from a Q-target network updated using a weakly augmented network. No Pretrain is much higher variance across the seeds and not able to match AugCL's performance on the walker walk task, thus showing the importance of first regularizing the CNN on weak augmentation, which motivates the curriculum. While Single Critic performs much better than No Pretrain, we can see it's much less sample efficient and converges to a lower solution

than AugCL on both the train environment and the test environment. We can see that while Single Critic is improving the policy it is learning it is does not perform as well as distributing learning the state, action value to the weakly augmented network, thus showing the non-zero destabilization strong augmentation incurs.

We also experimented with setting $M = 0$ and found it was unable to learn as the strong critic couldn't learn a useful representation and could not bootstrap the target network's predictions to improve learning. We include further exploration of selecting M in Sec. 6.2.5. This was indicated to us by the episodic reward not improving as training continued. We believe this also points towards the importance of weak augmentation regularization early in training. We believe that these experiments are clear evidence that strong augmentations do indeed incur a non-zero degradation to policy learning and that the key to getting good generalization performance is to disentangle the strong and weak augmentation as AugCL does, which is not possible without the curriculum as an unregularized CNN has difficulty learning task-relevant representations.

6.2.4 AugCL Train Environment Performance

We believe a key aspect of generalization includes maintaining the best policy possible on the train environment as well. We include in Tab. 6.5 train environment performance of AugCL across different strong augmentations we benchmarked on. We also include results for non-naive shift on the train environment as well as an upper bound to what all the generalized methods can achieve in Tab. 6.6.

Table 6.5: Train environment performance at the end of 500,000 train steps of AugCL with varying strong augmentations. The mean and standard deviation over 5 seeds where each seed is evaluated using the mean of 30 episodes done for each seed.

Strong Aug	Walker, Walk	Walker, Stand	Ball in Cup, Catch	Cartpole Swingup	Finger, Spin
Conv	894 ± 36	958 ± 13	965 ± 6	853 ± 6	979 ± 9
Overlay	903 ± 26	969 ± 7	964 ± 5	869 ± 10	967 ± 9
Splice	878 ± 38	958 ± 7	962 ± 6	865 ± 13	976 ± 15

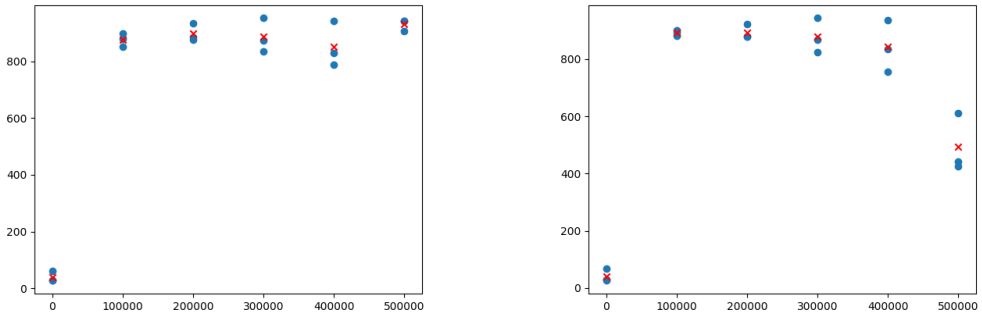
6.2.5 Choice Of M On Performance

We explored how different M selections effect AugCL in figure 6.4. We found a parabolic relationship between M and average episodic reward. We see on the test environment the relationship between M and test environment performance is parabolic, with performance peaking at M=100k or 200k on the test environment. While it seemed that

Table 6.6: Shift augmentation evaluation results on the train environment across 5 seeds with the mean and standard deviation across 30 episodes for each seed. This table serves as an upper bound to what can be achieved in generalized benchmarks.

Walker, Walk	Walker, Stand	Ball in Cup, Catch	Cartpole Swingup	Finger, Spin
916 ± 25	975 ± 2	971 ± 5	869 ± 11	984 ± 2

100k and 200k for M gave the same performance higher and lower values of M had much lower average performance. We theorize this is due to striking a good balance between regularizing the CNN with weak augmentation and then training it to adapt to the strongly augmented version of the environment, which requires a lot of frames to approximate.



(a) Train

(b) DMC-GB color hard

Figure 6.4: We show the effects of M selection on AugCL by choosing [0, 100k, 200k, 300k, 400k, 500k]. AugCL is trained with random convolution as the strong augmentation on Walker, Walk for a total of 500k frames. Blue circles represent the performance of different seeds using the average episodic reward over 50 runs and the red x represents the mean across all 3 seeds.

6.3 Summary

AugCL shows improvements in generalized environments by disentangling strong and weak augmentations into their respective networks. The combination of AugCL and Splice has substantially improved performance on DMC-GB, giving a new SOTA. We also effectively show the importance of weak augmented pre-training for parallel weak and strong augmented network training, highlighting the missing ingredients to previous attempts. An issue with our method is selecting the optimal M is still an open question. $M = 100,000$ yielded much worse results and we theorized that the CNN was not

regularized enough. This tricky balance can lead to significant changes in results and we hope to find a more developed method for selecting M.

Chapter 7 Conclusion

We have identified three issues with CNN architectures. **1)** When optimized using Cross-Entropy, $\|x\|_2$ is unbounded. **2)** $\|x\|_2$ has no correlation to accuracy. **3)** High-frequency feature bias of CNNs inherently causes RL agents to converge to lesser policies than when combined with weak augmentation. Both Ch. 5 and Ch. 6 use regularization in differing ways where Geometric Similarity Decomposition relies on a Temperature Scaling (22) form of regularization to resolve $\|x\|_2$ and a decomposed Cross-Entropy loss to increase sensitivity and Augmentation Curriculum Learning using shift as a weak augmentation to regularize the CNN to reduce bias towards high-frequency features.

An issue with Geometric Similarity Decomposition is that it decreases accuracy while increasing sensitivity, we found this is because it lowers the sensitivity of the class weights and therefore increases entropy. Increased entropy means more uncertainty which is in line to achieve better sensitivity, but also causes classification to be less accurate. For future works, it'd be important to achieve calibration and classification purely on the cosine similarity as it was shown to have a strong correlation to accuracy as shown in (13).

RL generalization methods using augmentation including Augmentation Curriculum Learning lack robustness to a set of visual perturbations. As shown in Ch. 6, it seems training under one augmentation only creates robustness to visually similar perturbations. This is not sufficient for robotics tasks such as autonomous vehicles, where differing weather and lighting situations can be seen as different visual perturbations (snow, rain, sunny day, evening). An investigation into the effects of augmentation on CNNs and how that improves robustness would greatly contribute to a more general approach to this problem. (17) showed a combination of augmentation can help with generalization, but the limits of this are still unexplored and real-world deployment tends to be might higher visual variance than simulation.

While our works here are a step forward towards better calibration CNNs and

generalization in RL. CNNs have lost favor since the release of Vision Transformers (16). These attention-based models still require much more exploration in terms of calibration and robustness, but they have been shown to overfit to the training environment in RL and do require strong augmentation as a way to mitigate this overfitting (25). Since GSD is applied to the final linear layer which still exists in Vision Transformers, it'll be interesting to see if Vision Transformers still suffer from calibration errors and if GSD is applicable as well as if AugCL improves performance on distribution-shifted environments.

Bibliography

[1] Activation functions in neural networks — by sagar sharma — towards data science. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. (Accessed on 11/30/2022).

[2] Cifar-10 and cifar-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html. (Accessed on 12/19/2022).

[3] Pooling (cnn) — epynn 1.0 documentation. https://epynn.net/Pooling.html. (Accessed on 12/02/2022).

[4] What are convolutional neural networks? — ibm. https://www.ibm.com/cloud/learn/convolutional-neural-networks. (Accessed on 11/30/2022).

[5] BALAJI, S. Binary image classifier cnn using tensorflow — by sai balaji — techiepedia — medium. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697. (Accessed on 12/02/2022).

[6] BENGIO, Y., LOURADOUR, J., COLLOBERT, R., AND WESTON, J. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ICML '09, Association for Computing Machinery, p. 41–48.

[7] BHARATH RAMSUNDAR, R. B. Z. 4. fully connected deep networks - tensorflow for deep learning [book]. https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html. (Accessed on 11/30/2022).

[8] BHATT, S. 5 things you need to know about reinforcement learning - kdnuggets.

https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html. (Accessed on 12/16/2022).

[9] BLACKBURN. Introduction to reinforcement learningnbsp;: Markov-decision process, May 2022.

[10] BRECHTEL, S., GINDELE, T., AND DILLMANN, R. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *17th international IEEE conference on intelligent transportation systems (ITSC)* (2014), IEEE, pp. 392–399.

[11] BRIER, G. W. Verification of forecasts expressed in terms of probability. *Monthly weather review 78*, 1 (1950), 1–3.

[12] CETIN, E., BALL, P. J., ROBERTS, S., AND CELIKTUTAN, O. Stabilizing off-policy deep reinforcement learning from pixels. In *International Conference on Machine Learning* (2022), PMLR, pp. 2784–2810.

[13] CHEN, B., LIU, W., YU, Z., KAUTZ, J., SHRIVASTAVA, A., GARG, A., AND ANANDKUMAR, A. Angular visual hardness. In *International Conference on Machine Learning* (2020), PMLR, pp. 1637–1648.

[14] COBBE, K., HESSE, C., HILTON, J., AND SCHULMAN, J. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588* (2019).

[15] DEGRIS, T., WHITE, M., AND SUTTON, R. S. Off-policy actor-critic, 2012.

[16] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., USZKOREIT, J., AND HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[17] FAN, L., WANG, G., HUANG, D.-A., YU, Z., FEI-FEI, L., ZHU, Y., AND ANANDKUMAR, A. Secant: Self-expert cloning for zero-shot generalization of visual policies. In *Proceedings of the 38th International Conference on Machine Learning* (18–24 Jul 2021), M. Meila and T. Zhang, Eds., vol. 139 of *Proceedings of Machine Learning Research*, PMLR, pp. 3088–3099.

[18] GAL, Y., AND GHAHRAMANI, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (2016), PMLR, pp. 1050–1059.

[19] GEIRHOS, R., RUBISCH, P., MICHAELIS, C., BETHGE, M., WICHMANN, F. A., AND BRENDEL, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2018.

[20] GERSHENSON, C. Artificial neural networks for beginners, 2003.

[21] GNEITING, T., AND RAFTERY, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association 102*, 477 (2007), 359–378.

[22] GUO, C., PLEISS, G., SUN, Y., AND WEINBERGER, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning* (2017), PMLR, pp. 1321–1330.

[23] HAARNOJA, T., ZHOU, A., ABBEEL, P., AND LEVINE, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

[24] HANSEN, N., JANGIR, R., SUN, Y., ALENYÀ, G., ABBEEL, P., EFROS, A. A., PINTO, L., AND WANG, X. Self-supervised policy adaptation during deployment. In *International Conference on Learning Representations* (2021).

[25] HANSEN, N., SU, H., AND WANG, X. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. In *Conference on Neural Information Processing Systems* (2021).

[26] HANSEN, N., AND WANG, X. Generalization in reinforcement learning by soft data augmentation. In *International Conference on Robotics and Automation* (2021).

[27] HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[28] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. *CoRR abs/1512.03385* (2015).

[29] HENDRYCKS, D., AND DIETTERICH, T. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations* (2019).

[30] HUANG, G., LIU, Z., VAN DER MAATEN, L., AND WEINBERGER, K. Q. Densely connected convolutional networks. In *CVPR* (2017), IEEE Computer Society, pp. 2261–2269.

[31] JO, J., AND BENGIO, Y. Measuring the tendency of cnns to learn surface statistical regularities, 2017.

[32] KALASHNIKOV, D., IRPAN, A., PASTOR, P., IBARZ, J., HERZOG, A., JANG, E., QUILLEN, D., HOLLY, E., KALAKRISHNAN, M., VANHOUCKE, V., AND LEVINE, S. Scalable deep reinforcement learning for vision-based robotic manipulation. In *2nd Annual Conference on Robot Learning, CoRL 2018, Zürich, Switzerland, 29-31 October 2018, Proceedings* (2018), vol. 87 of *Proceedings of Machine Learning Research*, PMLR, pp. 651–673.

[33] KANSIZOGLOU, I., BAMPIS, L., AND GASTERATOS, A. Deep feature space: A geometrical perspective. *arXiv preprint arXiv:2007.00062* (2020).

[34] KENDALL, A., AND GAL, Y. What uncertainties do we need in bayesian deep learning for computer vision? *arXiv preprint arXiv:1703.04977* (2017).

[35] KONDA, V., AND TSITSIKLIS, J. Actor-critic algorithms. In *Advances in Neural Information Processing Systems* (1999), S. Solla, T. Leen, and K. Müller, Eds., vol. 12, MIT Press.

[36] KRIZHEVSKY, A., HINTON, G., ET AL. Learning multiple layers of features from tiny images.

[37] LAKSHMINARAYANAN, B., PRITZEL, A., AND BLUNDELL, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474* (2016).

[38] LASKIN, M., LEE, K., STOOKE, A., PINTO, L., ABBEEL, P., AND SRINIVAS, A. Reinforcement learming with augmented data. arXiv:2004.14990.

[39] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE* (1998), pp. 2278–2324.

[40] LECUN, Y., AND CORTES, C. MNIST handwritten digit database.

[41] LEVINE, S., FINN, C., DARRELL, T., AND ABBEEL, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research 17*, 39 (2016), 1–40.

[42] LIN, T.-Y., MAIRE, M., BELONGIE, S., BOURDEV, L., GIRSHICK, R., HAYS, J., PERONA, P., RAMANAN, D., ZITNICK, C. L., AND DOLLÁR, P. Microsoft coco: Common objects in context, 2014.

[43] LIU, J. Z., LIN, Z., PADHY, S., TRAN, D., BEDRAX-WEISS, T., AND LAKSHMINARAYANAN, B. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *arXiv preprint arXiv:2006.10108* (2020).

[44] LIU, W., LIU, Z., YU, Z., DAI, B., LIN, R., WANG, Y., REHG, J. M., AND SONG, L. Decoupled networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 2771–2779.

[45] LIU, W., WEN, Y., YU, Z., AND YANG, M. Large-margin softmax loss for convolutional neural networks. In *ICML* (2016).

[46] MCCULLOCH, W. S., AND PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[47] MIKOŁAJCZYK, A., AND GROCHOWSKI, M. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)* (2018), pp. 117–122.

[48] MNIH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLOU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning, 2013.

[49] MURPHY, A. H. A new vector partition of the probability score. *Journal of Applied Meteorology and Climatology 12*, 4 (1973), 595–600.

[50] NAEINI, M. P., COOPER, G., AND HAUSKRECHT, M. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2015).

[51] NETZER, Y., WANG, T., COATES, A., BISSACCO, A., WU, B., AND NG, A. Y. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems* (2011).

[52] OVADIA, Y., FERTIG, E., REN, J., NADO, Z., SCULLEY, D., NOWOZIN, S., DILLON, J. V., LAKSHMINARAYANAN, B., AND SNOEK, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *arXiv preprint arXiv:1906.02530* (2019).

[53] PEREZ, L., AND WANG, J. The effectiveness of data augmentation in image classification using deep learning, 2017.

[54] PINTO, L., ANDRYCHOWICZ, M., WELINDER, P., ZAREMBA, W., AND ABBEEL, P. Asymmetric actor critic for image-based robot learning, 2017.

[55] PLATT, J., ET AL. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers 10*, 3 (1999), 61–74.

[56] RAILEANU, R., GOLDSTEIN, M., YARATS, D., KOSTRIKOV, I., AND FERGUS, R. Automatic data augmentation for generalization in deep reinforcement learning. *arXiv preprint arXiv:2006.12862* (2020).

[57] RIBA, E., MISHKIN, D., PONSA, D., RUBLEE, E., AND BRADSKI, G. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (2020), pp. 3674–3683.

[58] ROHRER, B. How to convert an rgb image to grayscale. `https://e2eml.school/convert_rgb_to_grayscale.html`. (Accessed on 11/30/2022).

[59] SACHAN, A. Detailed guide to understand and implement resnets – cv-tricks.com. `https://cv-tricks.com/keras/understand-implement-resnets/#:`

~:text=ResNet%20uses%20Batch%20Normalization%20at,network%20from%20vanishing%20gradient%20problem. (Accessed on 12/08/2022).

[60] SOUDRY, D., HOFFER, E., NACSON, M. S., GUNASEKAR, S., AND SREBRO, N. The implicit bias of gradient descent on separable data. In *The Journal of Machine Learning Research* (2018), p. 2822–2878.

[61] STONE, A., RAMIREZ, O., KONOLIGE, K., AND JONSCHKOWSKI, R. The distracting control suite – a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722* (2021).

[62] SUTTON, R. S., AND BARTO, A. G. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018.

[63] TASSA, Y., DORON, Y., MULDAL, A., EREZ, T., LI, Y., CASAS, D. D. L., BUDDEN, D., ABDOLMALEKI, A., MEREL, J., LEFRANCQ, A., ET AL. Deepmind control suite. *arXiv preprint arXiv:1801.00690* (2018).

[64] TOBIN, J., FONG, R., RAY, A., SCHNEIDER, J., ZAREMBA, W., AND ABBEEL, P. Domain randomization for transferring deep neural networks from simulation to the real world, 2017.

[65] VAN AMERSFOORT, J., SMITH, L., TEH, Y. W., AND GAL, Y. Uncertainty estimation using a single deep deterministic neural network. In *International Conference on Machine Learning* (2020), PMLR, pp. 9690–9700.

[66] WIJMANS, E., KADIAN, A., MORCOS, A., LEE, S., ESSA, I., PARIKH, D., SAVVA, M., AND BATRA, D. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357* (2019).

[67] YANG, Y., MA, Z., NIE, F., CHANG, X., AND HAUPTMANN, A. G. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision 113*, 2 (2015), 113–127.

[68] YARATS, D., KOSTRIKOV, I., AND FERGUS, R. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations* (2021).

[69] YOON, C. In-depth review of soft actor-critic — by chris yoon — towards data science. https://towardsdatascience.com/in-depth-review-of-soft-actor-critic-91448aba63d4. (Accessed on 01/05/2023).

[70] ZAGORUYKO, S., AND KOMODAKIS, N. Wide residual networks. *CoRR abs/1605.07146* (2016).

[71] ZHANG, H., CISSE, M., DAUPHIN, Y. N., AND LOPEZ-PAZ, D. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412* (2017).

[72] ZHOU, B., LAPEDRIZA, A., KHOSLA, A., OLIVA, A., AND TORRALBA, A. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 40*, 6 (2018), 1452–1464.

[73] ZHOU, T., TUCKER, R., FLYNN, J., FYFFE, G., AND SNAVELY, N. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH* (2018).

[74] ÅSTRÖM, K. J. Optimal control of markov processes with incomplete state information i. In *ISSN 0022247X*, vol. 10. p. 174–205.