Chapter 1

Introduction

1.1 Self-supervised Learning

Increasing computational power has led to increasing rates of data generation. However, the vast majority of available data is unlabeled. To contextualize: every day, 350 million images are uploaded to Facebook, 95 million images are shared on Instagram, and 5 billion images are shared daily on Snapchat.

Labeled data is expensive, difficult to create and share, and limited in scope. Imagenet 2012, one of the largest publicly available labeled image datasets took 2.5 years to produce, and includes just 1000 categories out of WordNet's original 117000 synsets [10, 13].

Self-supervised learning aims to overcome these limitations by using unlabeled data in concert with labeled data. This allows us to leverage the large amount of cheap unlabeled data that is easily accessible, while getting the best possible performance out of a limited-size labeled dataset. The core idea is to use the unlabeled data to build strong learned representations within a model. The learned representations will distill the salient information from input data. This is then paired with a small amount of labeled data to build an effective model for downstream tasks like image classification.

Initial self-supervised learning models tried to minimize the disagreement between the output of two networks. In Cowan et al. [8], networks are used to map the acoustic sounds associated with different vowels to codebook vectors in the same n-dimensional space. The

objective function tries to make these codebook vectors agree. In Becker and Hinton [3], a network maps a patch of an image to a scalar value, and the objective function tries to maximize mutual information of the scalar representation of two adjacent image patches. The network was able to discover depth in random dot stereograms of curved surfaces on its own.

Generally, modern self-supervised learning models can be categorized into 4 groups: self-distilliation, deep metric learning, correlation analysis, and masked modeling [2]. Each category of model deals with the issue of mode collapse of the learned representation using a different technique. Mode collapse is when the model learns to map all images to the same representation, causing the model to be unusable as an encoder for downstream tasks.

Self-distillation models use asymmetric model architectures to accomplish this. Two different views of a singular datum are fed to two different networks, and are mapped to each other with a predictor network. Metric learning models aim to decrease distance between similar data while increasing distance between dissimilar data. This is usually implemented as increasing similarity between semantically transformed versions of a singular datum. Correlation analysis models work more directly on hidden unit representations without explicit positive or negative examples. They decorrelate hidden unit responses from each other while making the hidden unit responses invariant to similar data. Masked input models provide the network with a partially masked input, while the network attempts to fill in the masked portion with the correct value. Lately, this strategy has been used with great success in large language models, and has been used in vision models with masked autoencoders.

1.2 Log polar transform

The log polar transform is a core property of the human visual system [14]. During visual processing, the mapping from the visual field to the primary visual cortex can be approximated with log polar [1]. We use this mapping in a self-supervised learning context because we want to mimic the human visual system to better understand, explore, and duplicate the learning process

that humans go through that allows them to become highly effective few-shot learners.

The log polar transform has a number of properties that make it a strong candidate for self-supervised learning. Log polar projections have a highly compressed information density of compared to the Cartesian representation of the same image, allowing us to focus on different parts of the image. Additionally, the log polar transform turns rotation of an image into a vertical translation and scaling of an image into a horizontal translation. Paired with a convolutional neural network, this grants us equivariance to these transformations. Finally, the log polar transform leads to vastly different images when centered at different points. Self-supervised learning depends on learning an invariance to transformed views of the same image, so a more powerful transformation will improve self-supervised learning.

These properties make log polar an interesting transformation to explore, especially in the context of self-supervised learning. In this paper, we rigorously test multiple self-supervised models across a variety of datasets, architectures, and image transformation pipelines. Our primary focus for this investigation is the use of the log polar projection as a data augmentation method in self-supervised learning.

Chapter 2

Background

2.1 SimCLR

SimCLR is a deep metric learning model introduced by Chen et al. [5]. Like other contrastive learning techniques, it maximizes similarity of positive examples and minimizes similarity of negative examples to learn an internal representation robust to different inputs. It treats two semantically transformed views of the same input image as a positive pair, and all other input examples as negative pairs.

The model uses the encoder network f_θ to create a learned representation vector of the image. The representation vector is high-dimensional, with the original paper using 2048 dimensions. This representation vector is then projected to $\mathbb{R}^{256}$ with a 2 layer fully connected projector network. The use of a projector network stabilizes training and help the model learn robust representations. The projector network is discarded after training is completed, and the encoder network is used to produce learned representations. A diagram of the architecture is provided in Figure 2.1.

The original paper uses cosine similarity for the similarity function: $\mathbf{S}(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{||\mathbf{u}||||\mathbf{v}||}$. The objective function in SimCLR is architected as follows: given a batch X of N original images with transformed representations X_1, X_2, representations $H_1, H_2 \in \mathbb{R}^{2048}$, and projections $Z_1, Z_2 \in \mathbb{R}^{256}$, the model concatenates the projections in paired order to form Z with $2N$ vectors

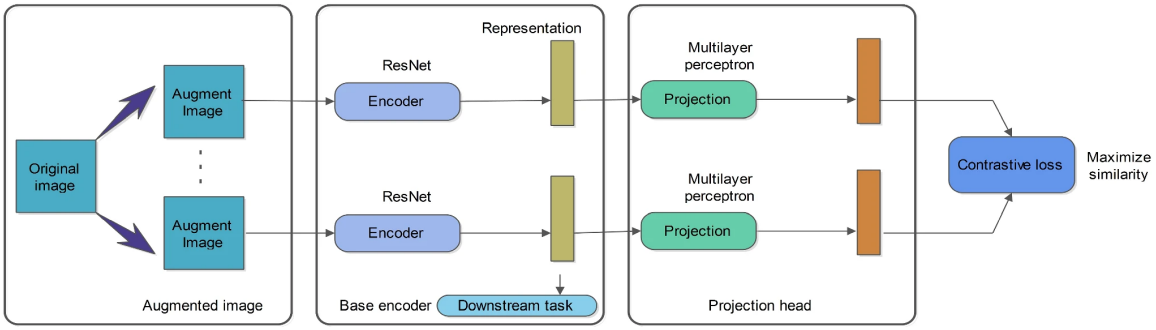

Figure 2.1. SimCLR Architecture Diagram [21]

such that $Z^{2k-1} = Z_1^k$ and $Z^{2k} = Z_2^k$. The model then computes intermediate value

$$l(i,j) = -\log \frac{\exp(\mathbf{S}(Z^i, Z^j)/\tau)}{\sum_{k=1;k\neq i}^{2N} \exp(\mathbf{S}(Z^i, Z^k)/\tau)}$$

The objective function seeks to minimize the quantity

$$\mathscr{L} = \frac{1}{2N} \sum_{i=1}^{N} (l(2i, 2i+1) + l(2i+1, 2i))$$

The result is to minimize the similarity of dissimilar pairs, while maximizing the similarity of similar pairs. The full algorithm pseudocode is provided in Algorithm 1.

The original paper operates with batch sizes of 8192. This dependency on large batches is a limiting factor of this class of models, where small batches do not contain enough negative examples to prevent mode collapse.

Prior work has shown that the image transformation pipeline has a large effect on the final performance of a self-supervised learning model [5, 17]. The original SimCLR paper uses a random image crop followed by a resize to the original image size, random color distortions including both color jitter and grayscaling, and random Gaussian blurring to generate the transformed images.

Algorithm 1. SimCLR's main learning algorithm

input: batch size N, softmax temperature τ, encoder network f, projector MLP g, transformation generator $\mathscr{T}$.

for sampled minibatch $\{x_k\}_{k=1}^{N}$ **do**

  for all $k \in \{1,\ldots,N\}$ **do**

  draw two augmentation functions $t \sim \mathscr{T}$, $t' \sim \mathscr{T}$

  $\tilde{x}_{2k-1} = t(x_k)$

  $h_{2k-1} = f(\tilde{x}_{2k-1})$

  $z_{2k-1} = g(h_{2k-1})$

  $\tilde{x}_{2k} = t'(x_k)$

  $h_{2k} = f(\tilde{x}_{2k})$

  $z_{2k} = g(h_{2k})$

  end for

  for all $i \in \{1,\ldots,2N\}$ and $j \in \{1,\ldots,2N\}$ **do**

    $s_{i,j} = z_i^\top z_j / (\|z_i\| \|z_j\|)$

  end for

  define $\ell(i,j)$ **as** $\ell(i,j) = -\log \frac{\exp(s_{i,j}/\tau)}{\sum_{k=1;k\neq i}^{2N} \exp(s_{i,k}/\tau)}$

  $\mathscr{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell(2k-1,2k) + \ell(2k,2k-1)]$

  update networks f and g to minimize $\mathscr{L}$

end for

return encoder network $f(\cdot)$, and throw away $g(\cdot)$

2.2 Bootstrap Your Own Latents (BYOL)

Bootstrap Your Own Latents (BYOL) [17], from the self-distillation family of models, does not rely on explicit negative pairs like SimCLR. Instead, it uses a student-teacher structure with two networks to minimize the risk of mode collapse. The teacher network is updated as an exponential moving average of the student network. This means the target network weights ξ are set as a function of themselves and the student network weights θ, controlled with a hyperparameter t for the rate of the moving average. This is implemented as $\xi = t\xi + (1-t)\theta$. In the original BYOL paper, t is initialized to 0.996 and linearly increased to 1 throughout training, leading to a very slow moving average. BYOL, like SimCLR, uses a fully-connected projector network, defined as g, immediately following the representational encoder. BYOL also adds a second fully-connected predictor network q, used to align the student projection with the teacher projection. The use of a slow moving exponential moving average generally prevents a collapsed representation. The full model architecture is visualized in Figure 2.2.

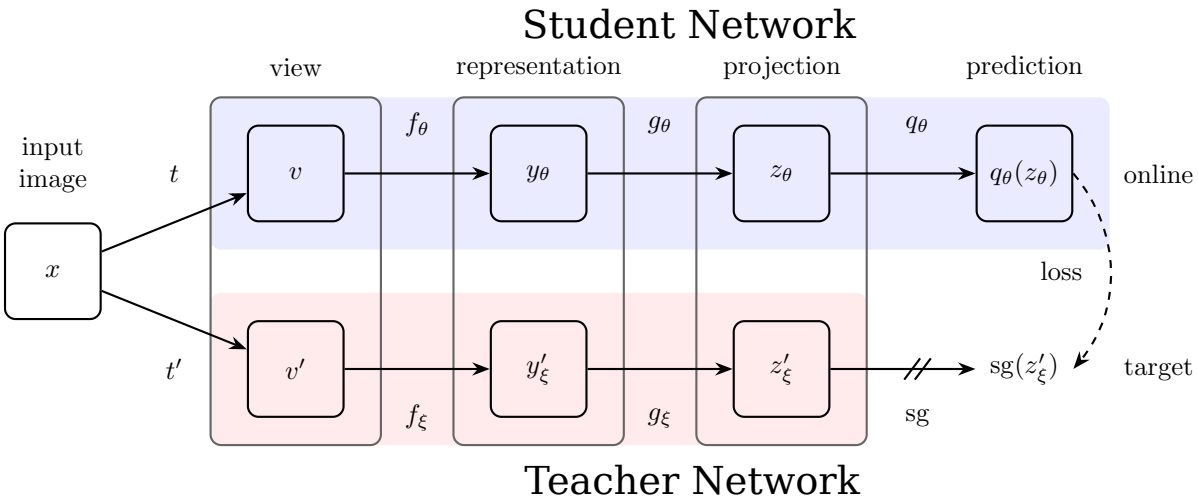

Figure 2.2. BYOL Architecture Diagram [17]

The objective function in BYOL is architected in the following manner: given a batch X of N original images with transformed representations X_1, X_2, the model computes $H_1 = f_\theta X_1$ and $H_2 = f_\xi X_2$ with the student and teacher networks respectively. From this, the model computes

$Z_1 = g_\theta(H_1)$ and $Z_2 = g_\xi(H_2)$ using the student and teacher projector MLP respectively, defined by g. The model applies a stop-gradient to Z_2, so no gradients are propagated back to any part of the teacher networks. Then, we compute $Q_1 = q_\theta(H_1)$. The model normalizes the vectors in Z_2 and Q_1. The model optimizes the final objective of $\mathscr{L} = ||\overline{Q_1} - \overline{Z_2}||_2^2$. The training algorithm is in Algorithm 2.

Algorithm 2. BYOL main learning algorithm [17]

input: batch size N, exponential moving average decay τ, encoder network f, projector MLP g, predictor MLP q, online parameters θ, target parameters ξ, output dimension d, transformation generator $\mathscr{T}$
for sampled minibatch $\{x_k\}_{k=1}^N$ **do**
  for all $k \in \{1,\ldots,N\}$ **do**
    draw two augmentation functions $t \sim \mathscr{T}$, $t' \sim \mathscr{T}$
    $z_1 = g_\theta\left(f_\theta(t(x_i))\right)$
    $z_2 = g_\theta\left(f_\theta(t'(x_i))\right)$
    $z_1' = g_\xi\left(f_\xi(t'(x_i))\right)$
    $z_2' = g_\xi\left(f_\xi(t(x_i))\right)$
    $l_i = -2 \cdot \left(\frac{\langle q_\theta(z_1), z_1' \rangle}{||q_\theta(z_1)||_2 \cdot ||z_1'||_2} + \frac{\langle q_\theta(z_2), z_2' \rangle}{||q_\theta(z_2)||_2 \cdot ||z_2'||_2}\right)$
  end for
  $\mathscr{L} = \sum_{i=1}^N l_i$
  update networks f_θ, g_θ, q_θ to minimize $\mathscr{L}$
  update networks f_ξ and g_ξ using exponential moving average parameterized by τ
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

BYOL presents two significant advantages over SimCLR. First, it can be applied to smaller batch sizes. Within the ablation studies performed in the original publication, BYOL had consistent performance over a wide range of batch sizes, only dropping significantly in accuracy at batch size 128 due to the batch normalization layer within the encoder CNN. Second, BYOL models are less dependent on the image transformation pipeline used. In the ablation studies, removing components from the image transformation pipeline caused BYOL's performance to drop significantly less than SimCLR with similar changes. This shows that BYOL is more robust to changes in the transformation pipeline. The specific changes can be seen in Figure 2.3.

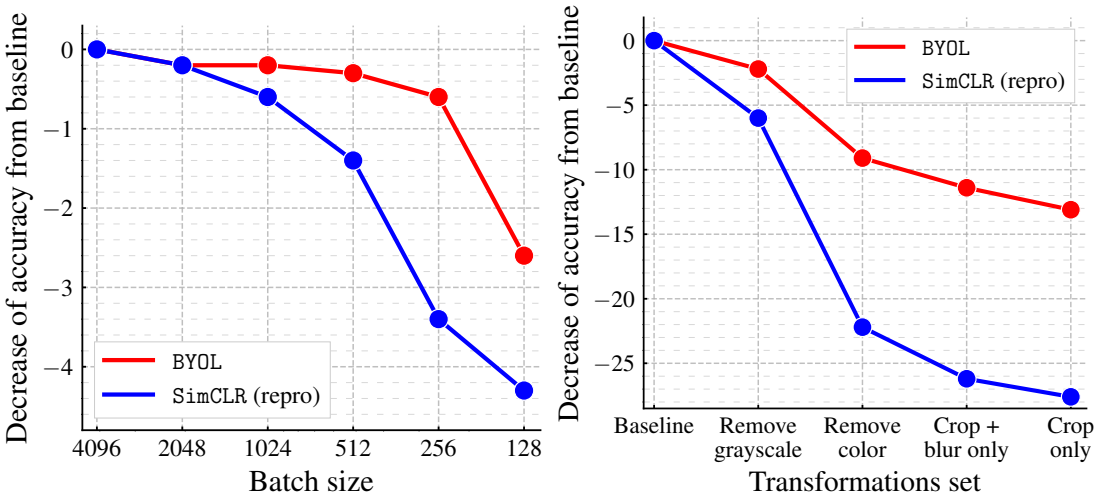

Figure 2.3. BYOL Ablation Studies Results [17]. Decrease in top-1 accuracy (in % points) of BYOL and reproduction of SimCLR at 300 epochs, under linear evaluation on ImageNet.

2.3 Barlow Twins

Barlow Twins is a non-contrastive learning technique from the correlation analysis family of models introduced in Zbontar et al. [34]. Barlow Twins attempts to minimize the covariance of different hidden units, while maximizing the invariance of a hidden unit to different transformations. This makes sure that each hidden unit is consistent between representations of the same image, while ensuring that different hidden units learn different features. Like SimCLR and BYOL, Barlow Twins uses a joint encoder-projector setup, where the projector is discarded after training. The full model architecture is visualized in Figure 2.4.

The objective function in Barlow Twins is architected in the following manner: given a batch X of N original images with transformed images X_1, X_2, compute $H_1 = f_\theta(X_1)$ and $H_2 = f_\theta(X_2)$ with the encoder. From this, compute $Z_1 = g_\theta(H_1)$ and $Z_2 = g_\theta(H_2)$ using the projector MLP. For stability, the model batch normalizes $\overline{Z_1} = \frac{Z_1 - \mu(Z_1)}{\sigma(Z_1)}$ and $\overline{Z_2} = \frac{Z_2 - \mu(Z_2)}{\sigma(Z_2)}$, where the mean and standard deviation are computed over the batch dimension. The model calculates the cross-correlation matrix C of $\overline{Z_1}$ and $\overline{Z_2}$, defined as $C_{i,j} = \sum_{k=1}^{N} \overline{Z_1}^{(k)} \overline{Z_2}^{(k)}$. The

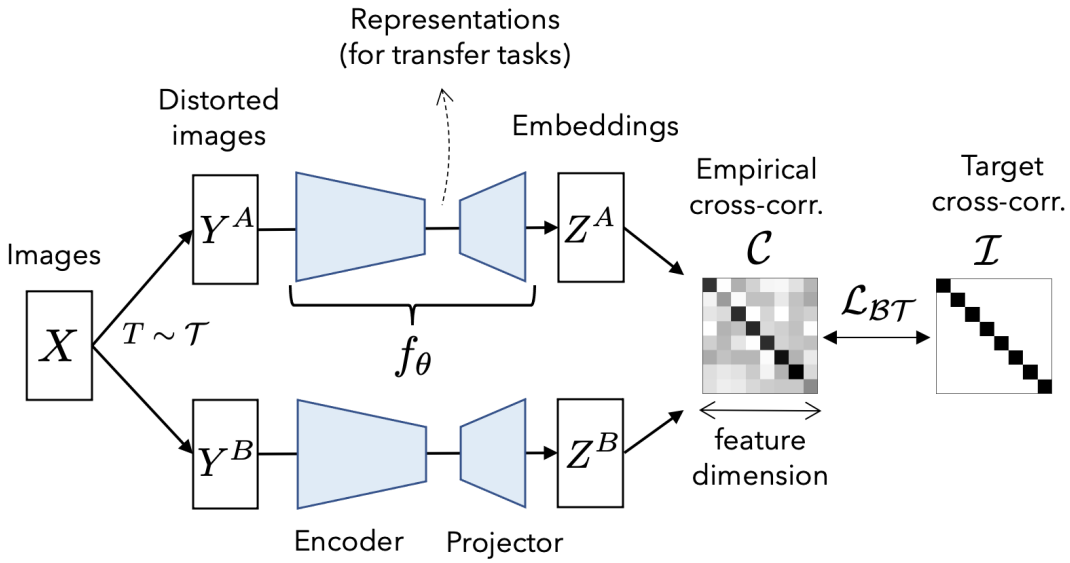

Figure 2.4. Barlow Twins architecture [34]

model optimizes the value

$$\mathscr{L}_{BT} = \sum_i (1 - C_{i,i})^2 + \lambda \sum_i \sum_{j \neq i} {C_{i,j}}^2$$

The full algorithm can be seen in Algorithm 3.

The first value is the invariance of the learned representation to different transformations, which the model maximizes during training. The second value, known as the redundancy-reduction term in the original work, is the correlation of different elements within the learned representation. Minimizing this value ensures that different hidden units learn different features. The calculation of the cross-correlation matrix does not follow the intuition used in other methods. Unlike deep metric learning techniques like SimCLR, the correlation matrix is calculated as similarity between two hidden units, rather than similarity between two images.

Barlow Twins models can use a smaller batch size than SimCLR without mode collapse, but it does not have the same robustness as BYOL. Barlow Twins requires a high-dimensional learned representation, with the original paper using 8192-dimension vectors, which is signifi-

cantly larger than the BYOL and SimCLR standard of 2048 dimensions. The higher dimensionality of the representation vectors requires more GPU memory to compute. As a result, per-GPU batch size has to be lowered to compensate.

Algorithm 3. Barlow Twins main learning algorithm [34]

input: batch size N, encoder network f, projector network g, output dimension d, transformation generator $\mathscr{T}$
for sampled minibatch $\{x_k\}_{k=1}^{N}$ **do**
  # Representation + Projection of minibatch
  for all $k \in \{1,\ldots,N\}$ **do**
    draw two augmentation functions $t \sim \mathscr{T}$, $t' \sim \mathscr{T}$
    $z_{k,1} = g_\theta(f_\theta(t(x_k)))$
    $z_{k,2} = g_\theta(f_\theta(t'(x_k)))$
  end for
  # Batch norm on projected representation
  $\overline{Z_1} = \frac{Z_1 - \mu(Z_1)}{\sigma(Z_1)}$
  $\overline{Z_2} = \frac{Z_2 - \mu(Z_2)}{\sigma(Z_2)}$
  # Correlation matrix and loss
  $C = \overline{Z_1}^T \overline{Z_2}$
  $\mathscr{L} = \sum_i (1 - C_{i,i})^2 + \lambda \sum_i \sum_{j \neq i} {C_{i,j}}^2$
  update networks f and g to minimize $\mathscr{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

2.4 Siamese Networks

Siamese networks were one of the original architectures to tackle semi-supervised learning introduced in Chopra et al. [7]. Unlike fully self-supervised learning models, original implementations of Siamese networks use labeled data. However, they do not use a traditional supervised classification objective like cross-entropy, but instead use a similarity objective function.

In their original implementations, Siamese networks increase the similarity of learned representations of all inputs of the same label. This is generally implemented in the following manner: given a batch X of N original images with labels Y, create transformed images X_1, X_2.

Compute $H_1 = f_\theta(X_1)$ and $H_2 = f_\theta(X_2)$ with the encoder network. With similarity function $\sim$, the network calculates $S_{i,j} = \sim(H_{1,i}, H_{2,j})$ The model optimizes the value

$$\mathscr{L}_{BT} = \sum_i \sum_j I(y_i, y_j) S_{i,j}^2 + (1 - I(y_i, y_j))(m - S_{i,j})^2$$

This algorithm depends on the labels to determine if two learned representations should be aligned or not. However, in the standard self-supervised training, this data is not available. The SimSiam algorithm corrects this using multiple transformed views of the same image.

SimSiam [6] maximizes the similarity between two augmentations of the same image. SimSiam is a deep metric learning model but it uses only one side of the metric learning framework - dropping negative examples to reduce computational load while maintaining performance. In this way, SimSiam is "BYOL without the momentum encoder" or "SimCLR without the negative examples".

SimSiam uses a stop gradient operation to prevent collapse. It is empirically shown that this is sufficient for preventing collapse. The architecture of the SimSiam model is described in Figure 2.5, and the full training algorithm is shown in Algorithm 4.

Algorithm 4. SimSiam main learning algorithm

input: batch size N, encoder network f, projector network g, predictor network h, output dimension d, transformation generator $\mathscr{T}$
for sampled minibatch $\{x_k\}_{k=1}^N$ **do**
  for all $k \in \{1, \ldots, N\}$ **do**
    draw two augmentation functions $t \sim \mathscr{T}$, $t' \sim \mathscr{T}$
    $z_{k,1} = g_\theta(f_\theta(t(x_k)))$
    $z_{k,2} = g_\theta(f_\theta(t'(x_k)))$
    $p_{k,1} = h_\theta(z_{k,1})$
    $p_{k,2} = h_\theta(z_{k,2})$
    $S_k = s(z_{k,1}, \texttt{stopgrad}(p_{k,2})) + s(z_{k,2}, \texttt{stopgrad}(p_{k,1}))$
  end for
  $\mathscr{L} = \frac{1}{2} \sum_k S_k$
  update networks f and g to minimize $\mathscr{L}$
end for
return encoder network $f(\cdot)$, and throw away $g(\cdot)$

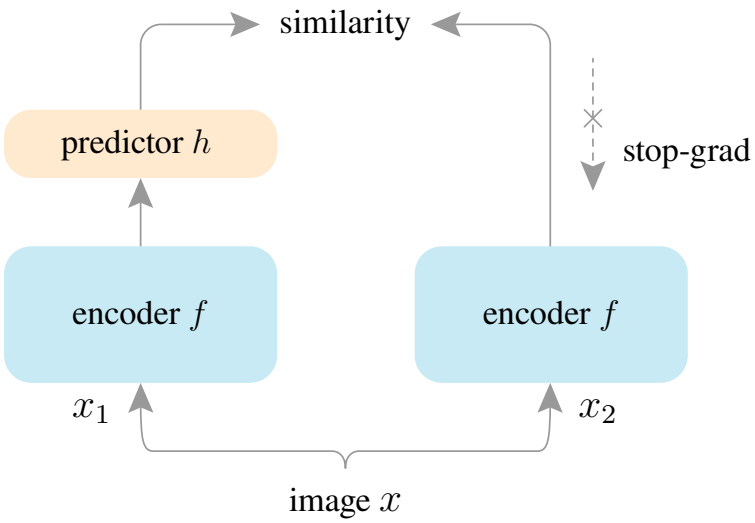

Figure 2.5. SimSiam architecture [6]

2.5 ResNet

The ResNet, or Residual Network, is a class of convolutional neural networks that have long chains of convolutional processing - up to 101 layers. Introduced in 2015 by [19], This is accomplished by reformulating the network to learn the residual function instead. This is implemented with skip connections, in which the output of a convolutional layer F is set as $G(x) = F(x) + x$. A diagram of the architecture of a layer is provided in Figure 2.6.

Under this algorithm, the network will learn to map $F(x) = G(x) - x$, the residual between $G(x)$ the optimal output and x the input. Learning the residual is an easier task than learning the original unreferenced function because the residual functions generally have smaller output ranges. This makes training more stable and less susceptible to gradient overflow or underflow.

The original ResNet paper presents models of varying depths between 18 and 101 layers. Reference papers in self-supervised learning generally use ResNet 50 as the backbone encoder for a balance of speed and parameter count.

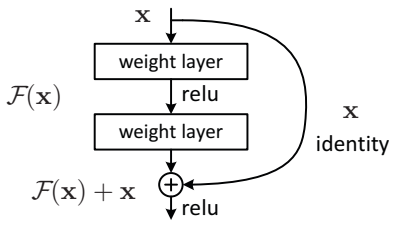

Figure 2.6. Layer architecture for learning residual functions in ResNet models

2.6 Log Polar Mapping

The log polar transform is a core part of human visual processing. During visual processing, the mapping from the visual field to the V1 visual cortex can be approximated with log polar [1]. Log polar transformations have been shown to be a good 2D approximation of the mapping of the visual field onto the visual cortex in primates and in computational models of facial recognition [15, 26, 14]. We incorporate the log polar mapping in our model so as to more closely approximate the signal processing in human vision. The log polar transform has a number of properties that would be intuitively beneficial to a self-supervised model.

The log polar transform produces a non-linear information density compared to the Cartesian representation of the same image. In a standard 224×224 image, the center 26×26 pixels will compose 50% of the final log polar image. In other words, the center 2% of the original image by area will compose 50% of the transformed image by pixel count; the center 25% of the original image will compose 93% of the transformed image. This allows us to focus on different parts of the image. This information density of the transformations we explore is visualized in Figure 2.7. The X-axis is the percentage of the original image, while the Y-axis is the percentage of the new transformed image.

Additionally, rotation in the Euclidean space becomes a vertical translation in the log polar space. Scaling an image down in Euclidean space becomes a horizontal translation in log polar space. This property is visualized in Figure 2.8. When paired with a shift-invariant network like a CNN, the log polar transform grants us rotational and scale invariance.

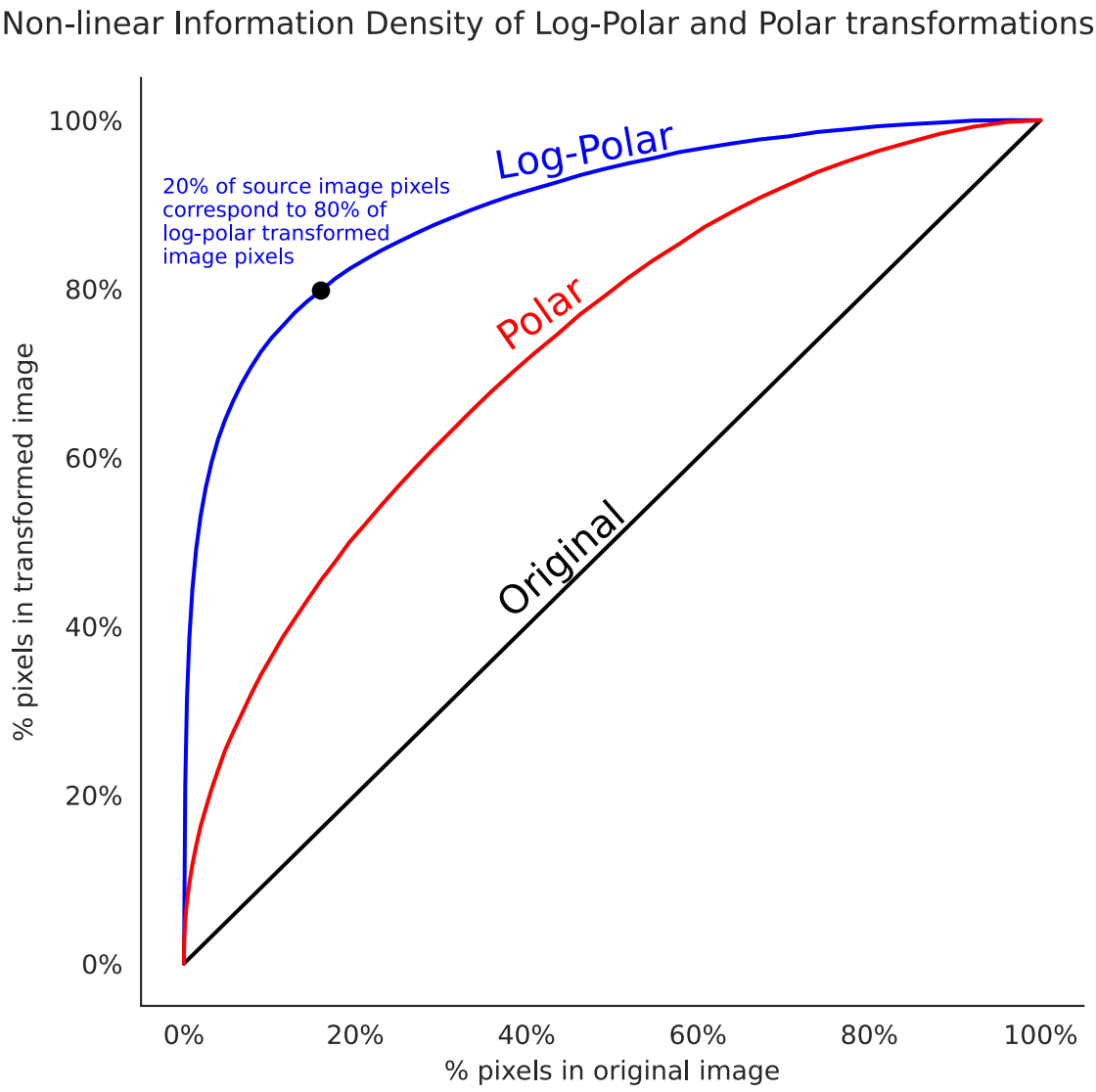

Figure 2.7. Non-linear Information Density of Transformed Images

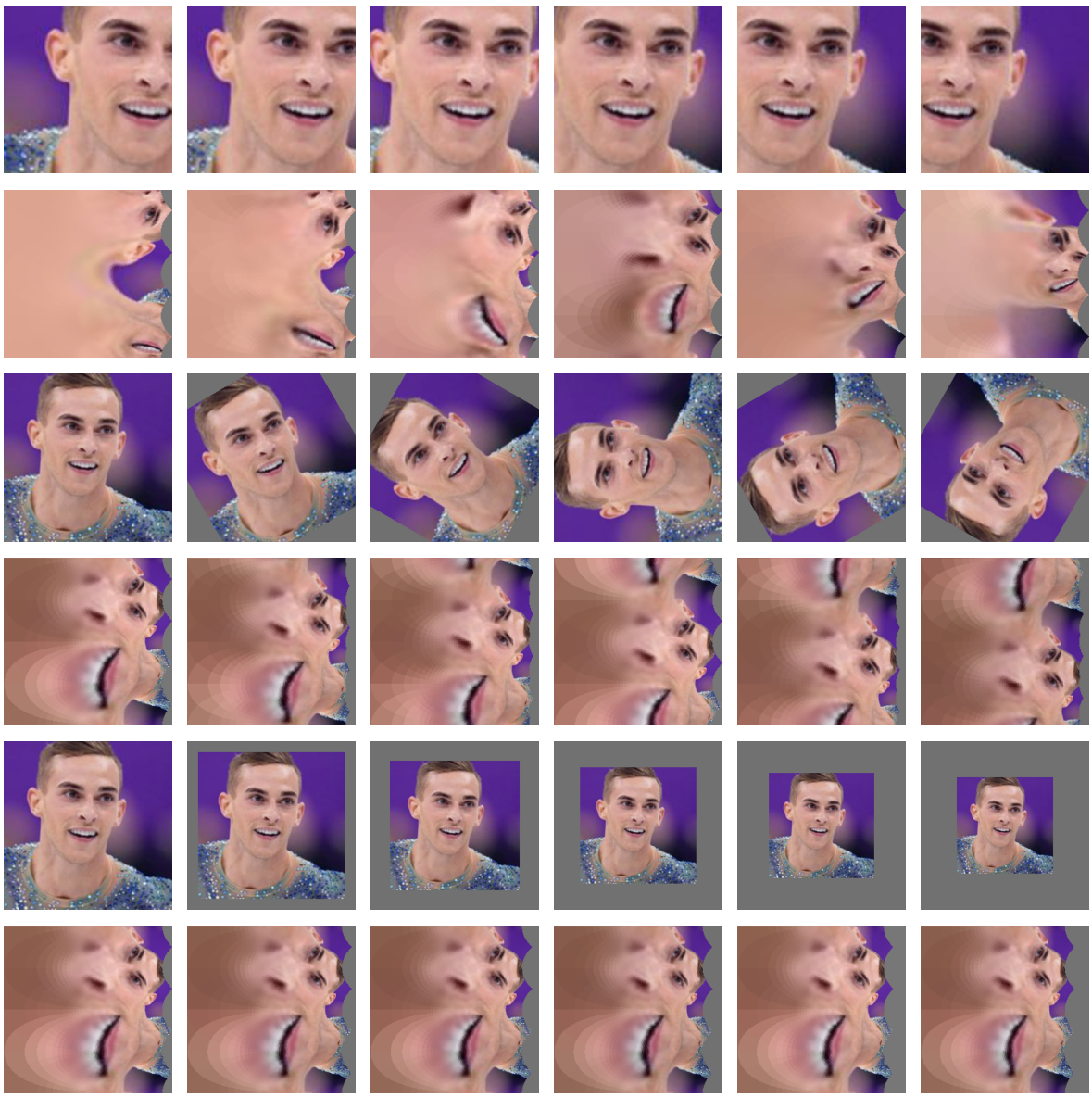

Figure 2.8. Projection to log polar coordinates turns rotation into vertical translation and scaling into horizontal translation. Translation causes disruption to the image.

Finally, the log polar transform leads to vastly different representations of the image when centered at different points. In Euclidean space, taking different crops of an image will still lead to generally similar images. The subject may be translated to different regions of the image, but a convolutional network has some inherent invariance to translations due to its architecture, which makes it generally robust to these transformations. As a result, unless the subject is partially obscured by translation, the transformed images are largely similar. However, when the log polar transform is applied, the transformed image changes significantly with small translations. We visualize this property in Figure 2.8. Self-supervised learning depends on the model learning invariance to transformed views of the same image, so a transformation that causes large visual differences should produce a more robust encoder network.

The use of the log polar transform in deep learning vision models has been explored in a variety of prior works. In Remmelzwaal et al. [28], the log polar transform is applied to image classification. The models are trained and tested on MNIST [23], which consists of 784 pixel grayscale images of handwritten digits. For training and testing, the images are transformed to log polar coordinate space, passed through a shallow 6-layer convolutional neural network, and classified with a 10 output softmax layer. The network uses no batch normalization or layer normalization, and uses max pooling to make the network invariant to small translations.

Remmelzwaal et al. [28] test both Euclidean and log polar coordinate models across a variety of image scaling and rotation. The log polar model generally outperforms the traditional CNN by up to 28% accuracy increase. We include a table showing the performance differences in Figure 2.9. The row indicates the amount of rotation, the column indicates the amount of scaling, and the cell value and color indicate the difference in performance between the log polar model and the Euclidean model. Green cells are rotation-scale combinations where the log polar model outperformed the Euclidean model, and red cells are rotation-scale combinations where the Euclidean model outperformed the log polar model.

Gahl et al. [15] applies the log polar transformation to image classification with more complex datasets. A standard ResNet-50 model is used with 4 different datasets - a faces dataset,

Accuracy improvement of (D) compared to (C)

Rotation	Scale 1.0	0.9	0.8	0.7	0.6	0.5	0.4
0	-1%	0%	1%	11%	20%	26%	8%
45	19%	19%	20%	22%	26%	24%	12%
90	21%	19%	16%	17%	13%	8%	6%
135	1%	-3%	7%	1%	10%	12%	0%
180	1%	3%	5%	-1%	6%	11%	0%
225	10%	9%	8%	6%	10%	10%	10%
270	12%	9%	6%	5%	3%	3%	0%
315	18%	23%	25%	28%	28%	27%	12%

Figure 2.9. Performance differences of log polar and Euclidean models on MNIST, various amounts of image rotation and scaling [28]

a dogs dataset, a cars dataset, and a dataset consisting of general objects that are always seen in an upright position (mono-oriented objects). This is paired with either Euclidean projection or the log polar projection. The paper looks specifically at the visual expertise effect, and how inversion affects classification accuracy. The model shows a similar performance on the validation set between both log polar and Euclidean models on upright images; log polar models have higher accuracy than Euclidean models when using inverted images.

In Cao et al. [4], the log polar transform is used with an image segmentation model. Feature maps are transformed to log polar coordinate space, segmented by a single shot detector [24] model with a 16 layer VGG [29] backbone, and inverse transformed to Euclidean space for loss and accuracy calculations. Cao et al. [4] also perform a study using multiple log polar projections of the same image for segmentation, each centered at a different location, similar to the human visual system's use of multiple fixation points during visual tasks [33]. The number of fixation points used is determined by the observation factor, or OF value.

The model is trained and tested on image segmentation tasks from the PASCAL VOC 2007 [11] and PASCAL VOC 2012 [12] datasets, and shows improvement over the state of the art baseline for segmentation R-CNN [16] in all cases. The log polar model shows a 3.4% mean average precision (mAP) increase over baseline models. When using rotated images, the model shows a 17.6% mAP increase over baseline. We include the results provided in Cao et al. [4] in

Figure 2.10.

Su and Wen [30] takes a different approach. Instead of transforming the original image to log polar space coordinates, it creates a new convolutional kernel that operates in log polar space. The standard convolution operator extracts a rectangular portion of an image, performs element-wise multiplication with parameter matrix W, and then sums the result. The log polar space convolution (LPSC) extracts a elliptical portion of an image, projects it to log polar space, and then performs the standard convolution operation (multiply then sum). The parameter matrix is learned in log polar space, while the image is kept in Euclidean space.

LPSC kernels are paired with a variety of models including AlexNet, VGG, and ResNet for models. The authors test these models on multiple datasets including CIFAR-10, CIFAR-100, Digital Retinal Images for Vessel Extraction (DRIVE), and Imagenet. LPSC models show higher performance across all datasets, although the increase is small in some cases. We include the reported performance in Table 2.1.

2.7 Hemispheric Transformation

As part of our investigation, we also mimic the human visual system hemispheric model of visual processing. During visual processing, the visual field is split into a left hemifield and a right hemifield, corresponding to visual input to the left and right eye [27]. During processing,

Table 2.1. Performance comparison of log polar space convolutions and standard convolutions, multiple datasets and models [30]

Dataset	Model	Original Accuracy (std-dev)	LPSC Accuracy (std-dev)
CIFAR-10	AlexNet	77.43 (0.25)	**78.44** (0.12)
CIFAR-10	ResNet-20	91.66 (0.13)	**91.81** (0.21)
CIFAR-10	VGG-19	93.54 (0.06)	**93.92** (0.06)
CIFAR-100	AlexNet	43.98 (0.43)	**47.43** (0.20)
CIFAR-100	VGG-19	72.41 (0.17)	**73.13** (0.12)
CIFAR-100	ResNet-20	67.56 (0.27)	**67.63** (0.27)
ImageNet	ResNet-18	69.86 (0.082)	**69.96** (0.081)

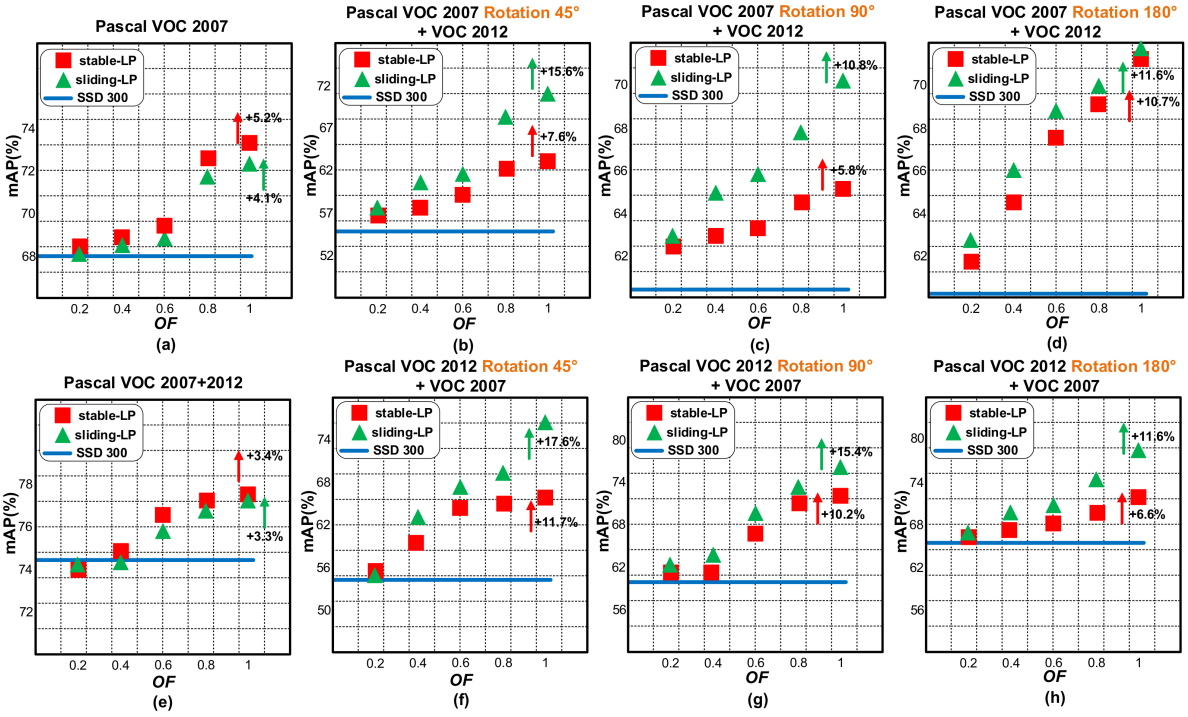

Figure 2.10. Log polar transformation confers an significant advantage for image segmentation tasks, which is magnified when using a higher observation factor (OF) with rotated images [4]

each brain hemisphere receives direct sensory information from the contralateral hemifield. Input from the left hemifield goes to the right hemisphere's visual cortex and input from the right hemifield goes to the left hemisphere's visual cortex. Creating a unified representation of of the complete visual field requires assimilating information initially projected to different halves of the brain [18].

Chapter 3

Methods

3.1 Data

We evaluated our models using three different training datasets. We began with the faces dataset. This consists of close up shots of 128 different people. This dataset was scraped by the GURU lab, and applied in Gahl et al. [14] and Gahl et al. [15]. We used a fixed train-validation-test split with all models of 17260 train images, 2159 validation images, and 2160 test images. The dataset split was created with stratified random sampling across each person. Example images from this dataset can be found in Figure 3.1.

Figure 3.1. Examples from Faces Dataset

The second dataset used is Imagenet Dogs, which is a subset of Imagenet2012. It consists of 118 breeds of dogs within the Imagenet dataset. This dataset is more challenging for networks to learn because the images are in a variety of orientations and poses. This is a larger dataset than the faces dataset with 134471 train images, 27498 validation images, and 27548 test images. Like the faces dataset, the split is created with stratified random sampling. Example images from this dataset can be found in Figure 3.2.

Figure 3.2. Examples from Dogs Dataset

Finally, we used the Imagenet 100 dataset. This is a randomly selected subset of 100 categories within Imagenet2012. We used this as a proxy for Imagenet evaluation in light of the computational demands of training and evaluating a model on the full scope of Imagenet. This is also a larger dataset than the faces dataset with 103296 train images, 12912 validation images, and 12950 test images. This dataset was created by taking a uniformly random sample without replacement of 100 categories of the Imagenet2012 dataset. Then, the dataset split is created with stratified random sampling across the selected categories. Example images from this dataset can be found in Figure 3.3.

The three different datasets require different degrees of visual expertise as a recognition task. Faces and dogs are subordinate categories that require fine grained discrimination. We need

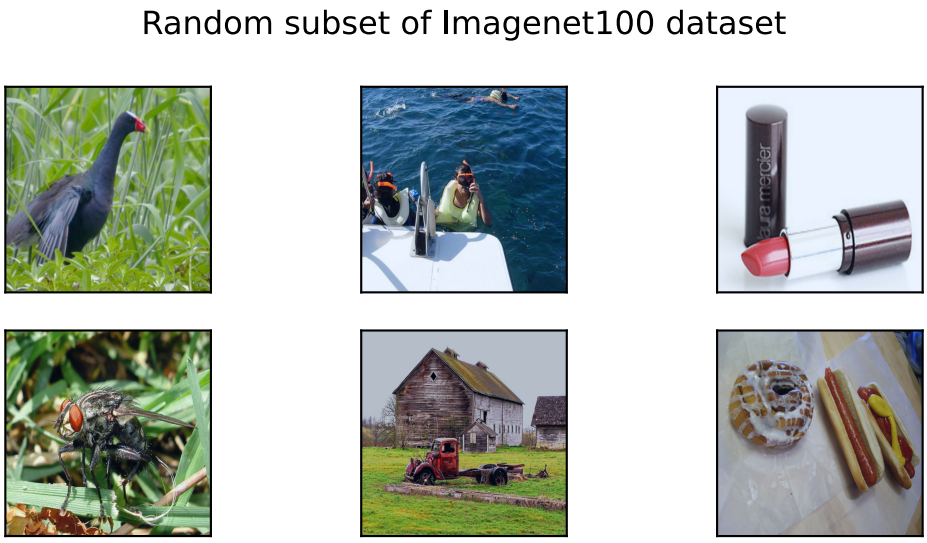

Figure 3.3. Examples from Imagenet100 Dataset. Categories from left to right, top to bottom are European gallinule, snorkel, lipstick, fly, barn, and hotdog

to be experts to distinguish faces, and studies have shown that we use configural information for facial recognition [14]. Configural processing means using relative measures like how far apart certain features are, as opposed to featural processing which deals with the existence of the feature. This is because the majority of faces have similar features (2 eyes, a nose, a mouth, etc.) for humans, but different faces have the eyes at different distances from each other or at different orientations to the rest of the face. In contrast, Imagenet100 consists of identifying basic level data. This means that the data is processed in a featural manner - with regards to the presence or absence of a certain feature. The question ”is this image a picture of a chair?” can generally be answered by examining the image for the presence of multiple legs and a seat. This is still a challenging question, since the basic-level category of ”chair” can include a variety of different types of chairs which look quite different from each other.

The faces dataset is a smaller dataset, with only 170 images per identity on average. The Imagenet subsets both have 1000+ images per category. For this reason, we focus our analysis on the latter two datasets primarily.

We use standard PyTorch dataloaders for our implementation. Transformations are

computed in parallel processes on the CPU, then final transformed images are moved to the GPU. We explored the use of FFCV [22], which is a just-in-time compiled transformation pipeline meant to be an optimized replacement for a standard PyTorch dataloader. However, this proved to be a slower dataloader due to a disk loading speed bottleneck. We also experimented with running transformations on the GPU, but this led to a bottleneck with data loading due to the size of the images.

3.2 Transformations

We use four different transformation pipelines within our experiments. The first pipeline is a copy of the SimCLR pipeline. This pipeline uses random crops, horizontal flip, color jitter, color grayscale, and random Gaussian blurring. We use this pipeline unchanged from the initial reference implementation.

We also replicate the pipeline described in Gahl et al. [15]. The Expertise pipeline is very similar to the SimCLR pipeline, except it replaces Gaussian blurring with a random rotation after the color jitter. For our purposes, we use a random rotation of up to 45 degrees with this pipeline.

The third pipeline is inspired by the hemispheric property of the human visual system. This pipeline splits a given image into two non-overlapping subimages, similar to the two hemifields that make up the complete visual field. Each half of the image is treated independently, similarly to how the hemispheres of the brain process their respective sensory inputs independently. The self-supervised model attempts to make the two subimages agree, inspired by how we believe the brain aligns the representations of the two different hemispheres to agree on what it is seeing.

Finally, we use a modified SimCLR pipeline that uses a log polar projection directly on the source image. For the first three pipelines, we use the log polar projection of an image after we crop it. For this pipeline, we replace the crop-then-project with a direct projection. For the standard Cartesian projection, we keep the default random resized crop used by SimCLR. In

the log polar projection, we replace the random resized crop with the log polar projection onto the original crop size. We introduce randomness into this otherwise deterministic projection by using randomly selected center points to fixate for the transformation. This provides the log polar projection with higher resolution source images, which helps improve the final representation quality and reduces visual artifacts in the final projection.

Examples of images created by the four pipelines are provided in Figure 3.4 and 3.5. Both figures show the original image in Euclidean or log polar space, as well as five transformations of that image using each of the different pipelines. The source image used is the same in both figures. We can see that the modified SimCLR pipeline has stark differences between transformations. Since the log polar module has a wider range of randomly selected center points for projection in this pipeline, the output images have more significant differences between them. The Expertise and SimCLR pipelines do not show as much of a difference, since they generally focus somewhere on the face.

For the majority of our transformations, we use standard PyTorch Torchvision library reference implementations. However, the library does not contain implementations of polar projection, log polar projection, or foveation. We use the implementation of foveation described in Perry and Geisler [25]. We reimplement polar and log polar projections as described in Thunuguntla [31], Zheng and Matungka [35], and van der Walt et al. [32] using CUDA compatible PyTorch code, making this a GPU-friendly transformation; this module is publicly available within our GitLab repository.

Since the log polar and polar transformations are non-linear mappings of Cartesian space, we see a variety of artifacts within the transformed image. We use multiple strategies to counter this and improve the quality of the transformed images. Firstly, we use a mask to crop out any pixels of the transformed image that have a preimage that are outside the bounds of the original image are discarded and masked out. Secondly, we use both nearest-neighbor and a distance-based approach to smoothing similar to bilinear smoothing. Finally, we experiment with both circumscribed and inscribed projections of the image. Since the final shape of a polar

or log polar projection is circular, we can either choose to inscribe or circumscribe the circular projection within the preimage. This shows a more significant impact when we use non-square source images such as with the modified SimCLR pipeline.

For Euclidean projection experiments, we use a crop size of 180 by 180 pixels. For log polar and polar projection experiments, we project to an output of 190 by 165 pixels. This roughly maintains the same number of total pixels (approximately 32000 pixels), while also having a similar aspect ratio as the binocular vision of the human visual system [9].

3.3 Models

We test SimCLR, BYOL, Barlow Twins, SimSiam, a supervised Siamese network, and a self-supervised Siamese network. With the exception of the supervised Siamese network, no labels are used during training. All models are trained with a batch size of 2048, sharded across 8 nodes with 256 images each. The encoder backbone is standardized to ResNet 50 to be consistent with the original papers for the tested models. The final embedding dimension is 2048, with the exception of Barlow Twins which uses an embedding of size 8192.

All models are trained with the Adam optimizer with an initial learning rate of 1e-4 and default betas of $(0.99, 0.999)$. We use a cosine annealing learning rate scheduler for warm restarts throughout training with a linear warmup of 10 epochs. We train for 1000 epochs with each model, according to findings in [17]. This is equivalent to 9000 training steps on the faces dataset, 66000 steps on the Imagenet Dogs dataset, and 51000 steps on the Imagenet100 dataset. Due to the differing dataset sizes, we do not place a heavy focus on cross-dataset evaluation.

3.4 Training

Compute is provided by NRP Nautilus. All training is standardized on machines with 8 CPU cores, 24 GB of RAM, and one NVIDIA A10 GPU. Training uses mixed precision FP16 floating point in PyTorch Lightning. We use PyTorch version 1.10.0 with CUDA version 11.3

and CUDA compute capability 8.0. Tensor cores are used to speed up training. We use 8-node clusters during training. Distributed sampling was handled by PyTorch Lightning with fixed random seed of 42.

3.5 Evaluation

We evaluate models using a linear classifier head. The linear classifier is trained on the representation vectors of all training images. This is done by taking upright, original size images from the training set, encoding them with the trained ResNet model, and performing logistic regression against their correct class output. The linear classifier is trained to minimize cross-entropy loss with the Adam optimizer for 10000 epochs at a learning rate of 0.001. The linear classifier is then evaluated using the representation vectors of all validation and test images for average accuracy and cross-entropy loss. To test how a model performs under new conditions, we rotate and scale down the image by various amounts as hyperparameters in the evaluation process. We report results at various rotation and scaling points, including the original size upright images. We can then better understand how the model performs will out-of-distribution images.

Our evaluation setup is a deviation from the general evaluation paradigm for self-supervised learning models where the whole model is finetuned on only a small percentage of training images. Specifically, this setup differs in the following ways:

1. We do not finetune the entire model - we treat the ResNet backbone as a black box and only change the weights of the linear classifier head.

2. We use the full training set of images to train the linear classifier.

3. We train the linear classifier head on upright unscaled images, but evaluate on multiple rotations and scales.

We evaluate against 25 different rotated and scaled variants of each validation and testing

image. Rotation varies between 0 and 180 degrees (full inversion), and scale values between 0.5 (half-size) and 1.0 (original size). Examples are included in Figures 3.6 and 3.7.

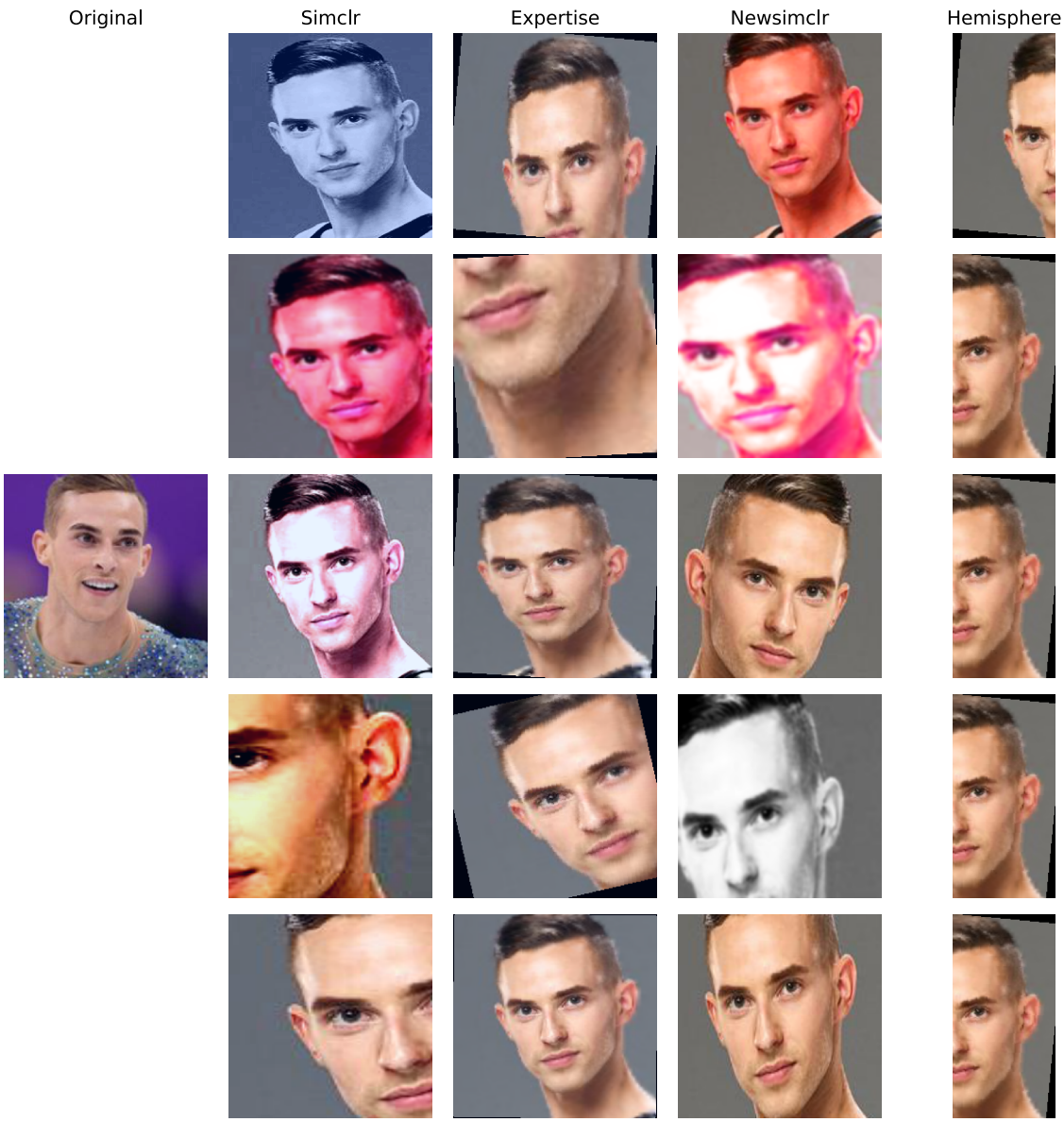

Figure 3.4. Euclidean projection examples of SimCLR, Expertise, Hemispheric, and modified SimCLR pipelines, applied to the same source image.

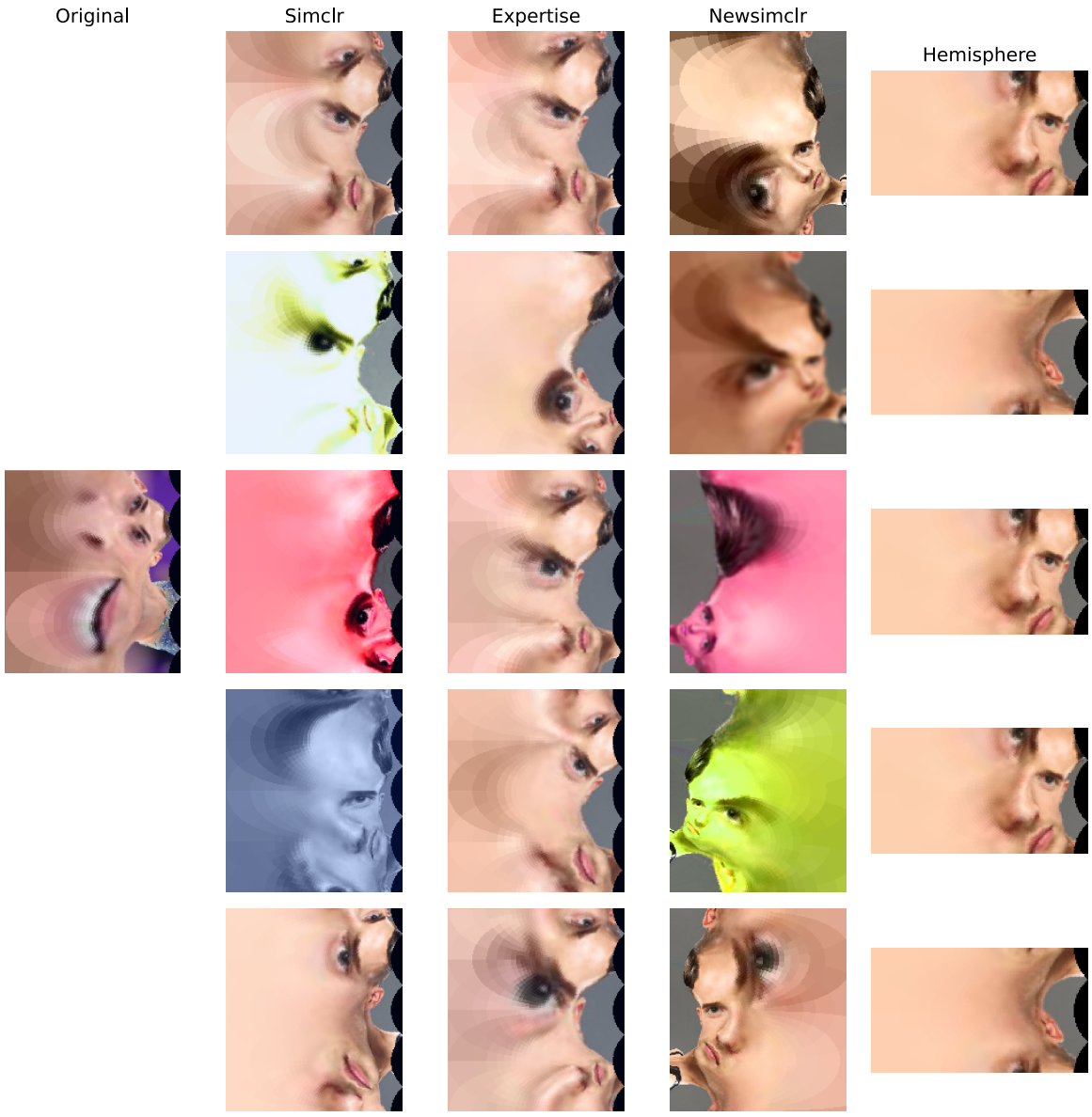

Figure 3.5. Log polar projection examples of SimCLR, Expertise, Hemispheric, and modified SimCLR pipelines, applied to the same source image.

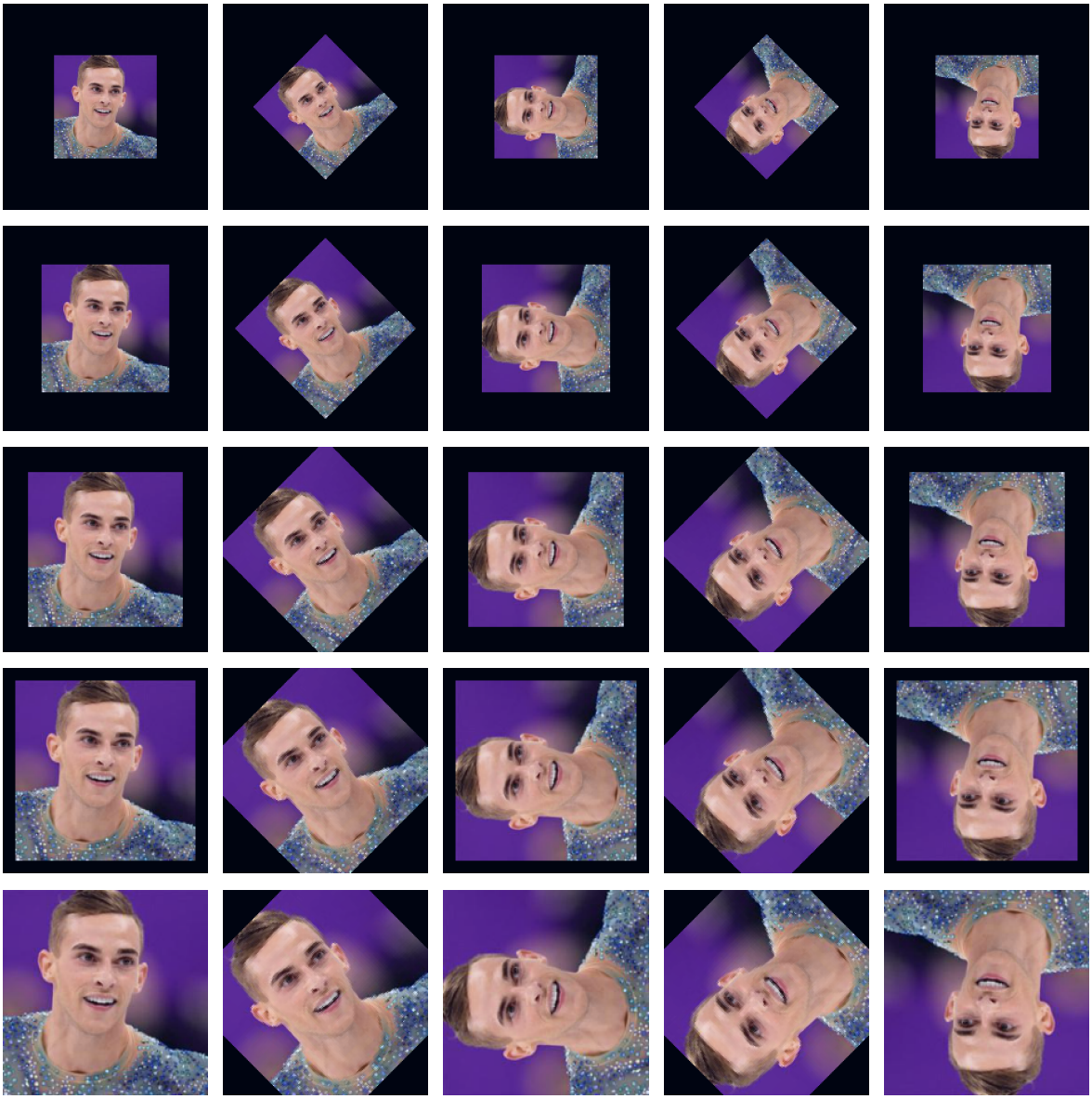

Figure 3.6. Rotated and scaled variants of validation images for out-of-distribution evaluation, Euclidean projection

.

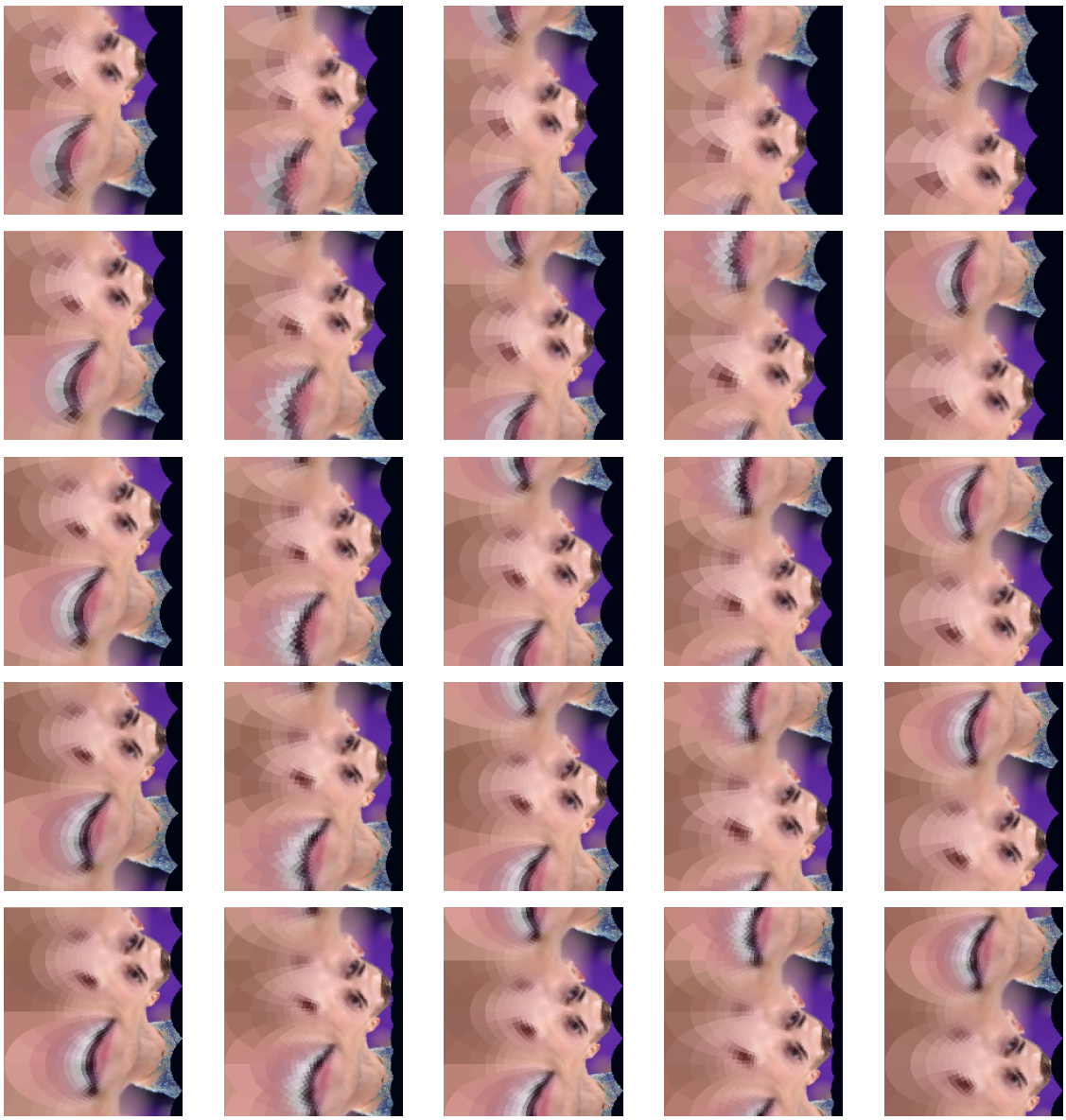

Figure 3.7. Rotated and scaled variants of validation images for out-of-distribution evaluation, log polar projection

Chapter 4

Results

We consider evaluation metrics in the context of different image transformation pipelines to understand how each model is affected by this choice. Each table includes results from one dataset on one model, evaluated at a variety of rotations and scales. The higher performing model at each rotation and scale is bolded for ease of comparison. Models were trained with one of the image transformation pipelines described in Section 3.2. At evaluation time, the model was fitted with a linear classifier head and trained on upright, original size images from the original training set as described in Section 3.5. The final accuracies are using the images from the testing set, rotating and scaling them as defined by evaluation hyperparameters, encoding them with the ResNet 50 backbone, and classifying them with the linear classifier. We focus on the performance of upright original size images, upright half-size images, inverted original size images, and inverted half-size images. In Figures 3.6 and 3.7, these are the images at the bottom left, top left, bottom right, and top right corners of the grid respectively.

4.1 Faces dataset

We start with the faces dataset. Results are presented in Tables 4.1, 4.2, and 4.3. Accompanying plots from training are presented in Section 5.1. Each table includes results from the faces data on one model. Models were trained with either the SimCLR or Expertise transformation pipeline. At evaluation time, the model was fitted with a linear classifier head.

Table 4.1. Performance comparison of log-polar and Euclidean projections, Faces dataset, BYOL Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**81.33**	74.88
SimCLR	Upright	Half-size	0.97	**25.07**
SimCLR	Inverted	Original	1.67	**18.58**
SimCLR	Inverted	Half-size	0.19	**5.47**
Expertise	Upright	Original	**74.24**	72.10
Expertise	Upright	Half-size	1.20	**1.67**
Expertise	Inverted	Original	4.68	**8.99**
Expertise	Inverted	Half-size	0.70	**1.30**

Table 4.2. Performance comparison of log-polar and Euclidean projections, Faces dataset, Barlow Twins Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	69.23	**71.22**
SimCLR	Upright	Half-size	1.58	**13.67**
SimCLR	Inverted	Original	2.83	**6.86**
SimCLR	Inverted	Half-size	0.97	**2.09**
Expertise	Upright	Original	**78.22**	73.54
Expertise	Upright	Half-size	2.41	**20.85**
Expertise	Inverted	Original	6.35	**17.84**
Expertise	Inverted	Half-size	1.39	**4.54**

In 5 out of 6 model-pipeline pairs with no rotation or scaling, the Euclidean projection model outperforms the log-polar projection model on the original upright images. The only instance where this does not happen is with the Barlow Twins model trained with the SimCLR transformation pipeline, where log-polar outperforms Euclidean by 1.99%. Generally, the Euclidean projection provides more information about upright, original-size images, leading to a higher linear classification accuracy. However, in 17 out of 18 model-pipeline pairs where we deal with images that were rotated or scaled at test time, log-polar models outperform their Euclidean counterparts. The only exception is the SimCLR model trained with the Expertise transformation pipeline, where the Euclidean model outperforms the log-polar model by 0.14%.

When we look at a model-level overview, we see that the Euclidean projection SimCLR

Table 4.3. Performance comparison of log-polar and Euclidean projections, Faces dataset, SimCLR Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**82.21**	79.15
SimCLR	Upright	Half-size	1.90	**13.67**
SimCLR	Inverted	Original	2.59	**11.72**
SimCLR	Inverted	Half-size	1.44	**3.24**
Expertise	Upright	Original	**68.81**	59.36
Expertise	Upright	Half-size	0.56	**1.58**
Expertise	Inverted	Original	2.92	**4.54**
Expertise	Inverted	Half-size	**0.65**	0.51

model is the most performant model, correctly classifying 82.21% of the upright, original sized test set. The log-polar projection SimCLR model is the most performant log-polar model, with a test accuracy of 79.15%. When we present the model with half-size inverted images, the log-polar space BYOL model is the most performant at 5.47% test accuracy, and the most performant Euclidean projection model is SimCLR at 1.44%.

4.2 Imagenet Dogs dataset

After evaluating the models on the faces dataset, we moved to the Imagenet Dogs dataset. This dataset is similar in composition to the faces dataset in that it focuses on subordinate level identification of dog breeds, which are subordinate categories of ”dogs” and all share similar traits. To distinguish dog breeds from each other requires the ability to do fine-grained discrimination of things like the texture and color of the coat, the style of the ears, or the relative size of the dog as measured by the length of their legs or torso. In this way, it is similar to the task of identifying the identity of human faces. However, this dataset is a much larger size. Due to this, we believe it is more indicative of the overall model's performance. Results can be found in Tables 4.4, 4.5, and 4.6.

We see that a similar pattern to the faces dataset holds, where log-polar space models underperform in the upright, original representation of the image, but generalize better to rotated

Table 4.4. Performance comparison of log-polar and Euclidean projections, Imagenet Dogs dataset, BYOL Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**71.97**	60.00
SimCLR	Upright	Half-size	19.77	**31.99**
SimCLR	Inverted	Original	18.52	**28.16**
SimCLR	Inverted	Half-size	3.81	**12.61**
Expertise	Upright	Original	**67.53**	52.55
Expertise	Upright	Half-size	5.02	**12.47**
Expertise	Inverted	Original	15.97	**23.26**
Expertise	Inverted	Half-size	2.44	**7.37**

Table 4.5. Performance comparison of log-polar and Euclidean projections, Imagenet Dogs dataset, Barlow Twins Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**67.70**	54.03
SimCLR	Upright	Half-size	11.80	**18.05**
SimCLR	Inverted	Original	11.47	**16.56**
SimCLR	Inverted	Half-size	2.82	**6.23**
Expertise	Upright	Original	**69.21**	55.72
Expertise	Upright	Half-size	12.14	**15.24**
Expertise	Inverted	Original	18.43	**25.79**
Expertise	Inverted	Half-size	4.09	**7.65**

or scaled representations of the image.

We also want to note that the dogs dataset appears to be more difficult for log-polar space models, since the performance drop on upright original-scale images is more than it was on the faces dataset. With the faces dataset, the performance drop on in-distribution images was less than 7%. With the dogs dataset, we see a drop of 18.28% in a Euclidean SimCLR model and log-polar SimCLR model, both trained with the same SimCLR pipeline.

The BYOL models trained with the SimCLR transformation pipeline have the best test accuracy on upright, original size images, at 71.97% for the Euclidean projection model and 60.00% for the log-polar projection model. For inverted, half-size images, the BYOL log-polar projection model has a test accuracy of 12.61% using the SimCLR transformation pipeline, and

Table 4.6. Performance comparison of log-polar and Euclidean projections, Imagenet Dogs dataset, SimCLR Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**70.52**	52.24
SimCLR	Upright	Half-size	5.67	**14.49**
SimCLR	Inverted	Original	13.51	**21.94**
SimCLR	Inverted	Half-size	2.09	**7.84**
Expertise	Upright	Original	**45.81**	36.11
Expertise	Upright	Half-size	1.57	**1.70**
Expertise	Inverted	Original	6.18	**11.05**
Expertise	Inverted	Half-size	1.06	**1.50**

the Barlow Twins Euclidean projection model has a test accuracy of 4.09% with the Expertise transformation pipeline.

Upright, original size classification accuracy has gone down with the dogs dataset compared to the faces dataset, while the classification accuracy of transformed images has gone up between the faces dataset and dogs dataset. The best accuracy on upright original size images went from 82.21% to 71.97% (a decrease of 10.24%). However, the best accuracy on upright half-size images went from 25.07% to 31.99% (an increase of 6.92%), inverted original size images went from 18.58% to 28.16% (an increase of 9.58%), and inverted half-size images went from 5.47% to 12.61% (an increase of 7.14%). We believe this is because the dogs dataset is harder to classify correctly due to its increased size and complexity since it presents images in a wider variety of poses, environments, and quality. However, this increaseed complexity allows models to generalize better to transformed images.

4.3 Imagenet100 Dataset

Our final dataset is the Imagenet100 dataset, a random sample of 100 Imagenet categories. We use the Imagenet100 dataset to compare fine grain discrimination models to a model trained on basic level categories. This allows us to test our models' abilities to do coarse-grained discrimination. Some of the categories in the Imagenet100 subset we include goldfish, golf

balls, and Model T cars. For the full list, consult Appendix 2. Images are seen in very different contexts and are comprised of very different shapes and textures. Due to that, the model should have an easier time on this dataset. This dataset should also provide a more representative view of real-world image classification where we often use basic-level categories to describe images. Results are presented in Tables 4.7, 4.8, and 4.9. Accompanying plots from training are presented in Section 5.1.

Table 4.7. Performance comparison of log-polar and Euclidean projections, Imagenet100 dataset, BYOL Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**60.11**	57.63
SimCLR	Upright	Half-size	5.86	**14.02**
SimCLR	Inverted	Original	26.79	**36.09**
SimCLR	Inverted	Half-size	3.24	**10.15**
Expertise	Upright	Original	**61.99**	59.24
Expertise	Upright	Half-size	8.32	**13.43**
Expertise	Inverted	Original	33.47	**42.91**
Expertise	Inverted	Half-size	4.93	**10.18**

Table 4.8. Performance comparison of log-polar and Euclidean projections, Imagenet100 dataset, Barlow Twins Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**65.44**	63.98
SimCLR	Upright	Half-size	**10.50**	9.33
SimCLR	Inverted	Original	29.74	**40.79**
SimCLR	Inverted	Half-size	5.58	**5.92**
Expertise	Upright	Original	66.39	**67.04**
Expertise	Upright	Half-size	10.36	**20.22**
Expertise	Inverted	Original	39.10	**50.02**
Expertise	Inverted	Half-size	7.41	**14.38**

This dataset shows generally similar results to the dogs dataset and faces dataset examined in prior sections. However, there are some specific, notable differences we point out and explain.

Generally, Euclidean projection models show a better test set accuracy on upright original size images, as compared to log polar projection models. However, the gap is much smaller than

Table 4.9. Performance comparison of log-polar and Euclidean projections, Imagenet100 dataset, SimCLR Model

Pipeline	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
SimCLR	Upright	Original	**63.30**	61.00
SimCLR	Upright	Half-size	7.62	**14.72**
SimCLR	Inverted	Original	28.73	**39.88**
SimCLR	Inverted	Half-size	5.10	**10.07**
Expertise	Upright	Original	**62.94**	61.61
Expertise	Upright	Half-size	9.76	**11.96**
Expertise	Inverted	Original	33.67	**43.15**
Expertise	Inverted	Half-size	6.75	**9.01**

the gap we saw on the dogs dataset. On this dataset, the gap is 3% in the worst case, and often times lower than this. The best classification accuracy is acheived by Barlow Twins trained with the Expertise pipeline in a log polar projection, correctly classifying 67% of the test set. This dataset is the only where the best performing model is a log polar projection model; on both the faces and dogs datasets, the highest accuracy was achieved by Euclidean projection models.

Additionally, the models perform better on rotated and scaled images. The performance drop with fully inverting images or scaling them to half size is lower. We can see up to 50% classification accuracy with inverted images, and up to 20% accuracy on half-size images. The accuracy drop from upright images to inverted images is 17%, compared to around 30% drop on the dogs dataset and 50% drop on the faces dataset.

Both of these properties are tied to the basic level categorization of this dataset compared to the finer grained discrimination skills needed on the dogs and faces dataset. The ability to generalize to inverted images has been shown to specifically be tied to the use of featural processing over configural processing [15]. However, it doesn't appear that this task is significantly easier than the other two - the best test set accuracy of 67% is close to the best test set accuracy on the dogs dataset. The properties of this dataset appear to allow for better generalization, as opposed to making the task easier.

4.4 Hemisphere Pipeline

We examine the performance of the hemispheric pipeline in Table 4.10. The pipeline is evaluated on all 3 datasets using the Barlow Twins model. This set was usually trained for considerably less than the 1000 epochs standardized for all other model-pipeline combinations, due to the model failing to improve. However, we report these partial results to discuss why this may have happened.

Table 4.10. Hemispheric pipeline performance, log polar and Euclidean projections, Barlow Twins model, various datasets

Dataset	Rotation	Scaling	Euclidean Acc. (%)	Log-Polar Acc. (%)
Imagenet Dogs	Upright	Original	**13.42**	6.83
Imagenet Dogs	Upright	Half-size	1.65	**2.99**
Imagenet Dogs	Inverted	Original	**5.18**	3.92
Imagenet Dogs	Inverted	Half-size	1.53	**2.69**
Faces	Upright	Original	19.97	**25.16**
Faces	Upright	Half-size	0.79	**1.39**
Faces	Inverted	Original	3.29	**7.14**
Faces	Inverted	Half-size	0.65	**0.74**
Imagenet100	Upright	Original	**22.45**	16.12
Imagenet100	Upright	Half-size	2.08	**3.78**
Imagenet100	Inverted	Original	11.17	**11.81**
Imagenet100	Inverted	Half-size	2.05	**3.74**

We can see that across the board, models trained with the hemispheric pipeline do not acheive the same test accuracy as the models trained with the SimCLR or Expertise pipelines on any of the three datasets. When we look at examples of transformed images created by the hemispheric pipeline in Figures 3.4 and 3.5, we can see that the images do not resemble each other very much since they come from completely distinct patches of the source image. However, the two hemifields of the human visual system show considerable overlap with each other, with 120 degrees of binocular vision [20]. A more efficient and appropriate hemispheric representation may take this into account, and present partially overlapping patches of the source image to the model. Aligning these may prove easier since there is some correspondence between the two

input images to the model.

4.5 Modified SimCLR Pipeline

We examine the performance of the modified SimCLR pipeline in Table 4.11. The pipeline is evaluated on the Imagenet100 dataset across the 3 models. In this experiment we only use the log polar projection, since the modified SimCLR pipeline is identical to the original SimCLR pipeline when using a Euclidean projection. We report both training and test set accuracy, to examine how this pipeline generalizes from the training set to the test set.

Table 4.11. Modified SimCLR pipeline performance on Imagenet100 dataset, log polar projection, training and test set

Model	Rotation	Scaling	Train Acc. (%)	Test Acc. (%)
SimCLR	Upright	Original	**39.84**	10.14
SimCLR	Upright	Half-size	**3.80**	3.47
SimCLR	Inverted	Original	**11.71**	11.17
SimCLR	Inverted	Half-size	2.84	**2.98**
BYOL	Upright	Original	**43.75**	12.66
BYOL	Upright	Half-size	**3.58**	3.44
BYOL	Inverted	Original	**11.22**	11.00
BYOL	Inverted	Half-size	2.90	**3.12**
Barlow Twins	Upright	Original	**36.72**	7.25
Barlow Twins	Upright	Half-size	**2.91**	2.67
Barlow Twins	Inverted	Original	**7.88**	7.63
Barlow Twins	Inverted	Half-size	2.40	**2.44**

If we only look at this pipeline's performance on the test set, it appears that all 3 models have failed to learn anything. However, when we consider training set accuracy, we see that the models can perform reasonably well on upright original size images. The performance is lower than that presented in 4.3, but better than the accuracy at the beginning of training. The issue is that the models are failing to generalize from the training set to the test set. When we look at the transformed images outputted by the modified SimCLR pipeline in Figure 3.5, we see that the images are very dissimilar to each other, despite being transformations of the same source image.

Due to this, the model can sometimes learn to align representations of the transformed training images, but fails to generalize to the test set since those transformed images have never been seen before.

Chapter 5

Analysis

In this chapter, we analyze various training and evaluation metrics in the context of the different architecture choices to understand how performance is affected by our decisions.

5.1 Training Plots

In Figures 5.1, 5.2, and 5.3, we present online evaluator accuracy over 1000 epochs of training. While training the network, we concurrently train a linear classifier over the encoded representation of the training set images. The linear classifier is trained using a supervised cross-entropy loss against the provided labels. This allows us to ensure models are successfully training and improving. Some models were not trained for the full 1000 epochs, either due to poor performance or instability within the training cluster.

Cool colors (blue, purple) are Euclidean projection models. Warm colors (red, orange) are log polar projection models. We don't see any instances of a log polar model showing a higher classification accuracy on the training or validation set. However, we do see some datasets having a smaller gap between log polar model accuracy and Euclidean model accuracy. Models trained on the Imagenet100 dataset generally end within 10% validation accuracy of each other. Models trained on the dogs dataset show a wider gap of 20% between log polar and Euclidean models trained with the same transformation pipeline.

The hemisphere pipeline (paired with Barlow Twins models) consistently fails to improve

on any of the three datasets in both Euclidean projections and log polar projections. When we examine the images created with this pipeline (see Figure 3.4 and 3.5), we see that the images are always half inputs - either left or right visual hemifield but never both. The model is always working with only half the information. Additionally, the images are completely distinct from each other - there is no overlap between the two visual hemifields. The combination of these two properties appears to make learning with a hemispheric pipeline too challenging for these models.

The modified SimCLR pipeline is only trained with a log polar projection on the Imagenet100 dataset. It's paired with Barlow Twins, BYOL, and SimCLR models. Here, we see a large drop between validation and training accuracy - a greater drop than any other model-pipeline pairs on this dataset. Upon examination of the images created by the modified SimCLR pipeline (see Figure 3.4 and 3.5), we see that different transformations of the same source image through this pipeline can vary a lot, depending on the random center point chosen. The stark difference between each random transformation of the same source image seems to mean that the network cannot learn to map all transformations of a single source image to the same representation for the online evaluator to use. This leads to the online evaluator not being able to generalize between the training and validation set.

5.2 Heatmaps of Performance Change between Log Polar and Euclidean Models

To better understand the performance difference between log polar and Euclidean projection models, we build a set of heatmaps like Remmelzwaal et al. [28]. Green cells indicate combinations where log polar models outperform their correponding Euclidean counterparts. Refer to Figures 5.5, 5.4, and 5.6.

Across all of the generated heatmaps, there are only a few significant dark spots constrained to the top left corner of the figure. The log polar model underperforms the Euclidean

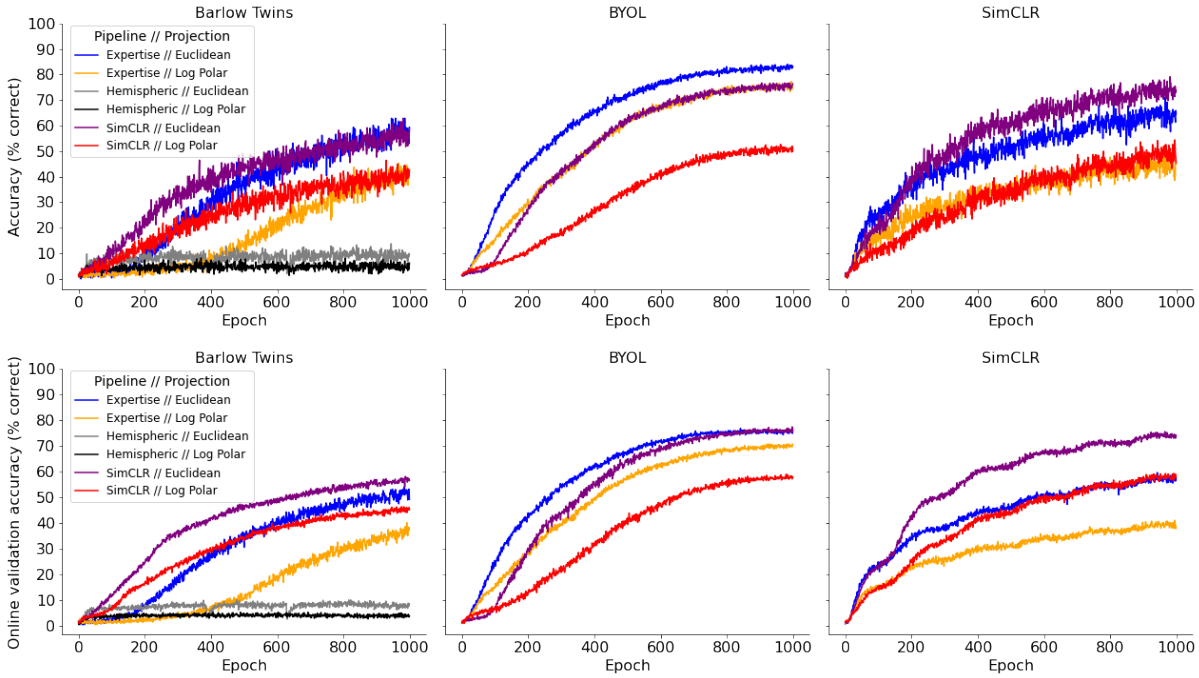

Figure 5.1. Faces dataset training plots - online evaluator accuracy on training set(top) and validation set (bottom), various models

model on upright, original scale images usually. However, if we rotate and scale down the image, log polar usually has at least some advantage over the Euclidean model.

The darkest green patches are where the log-polar model has a larger advantage. For most of the models we look at, this is in the lower left corner which is minor amounts of scaling but large amounts of rotation. The top right corner, where we scale down the image but don't rotate also performs well on some models, but this is less consistent.

SimCLR is the most consistent model class by performance gap. It very consistently has 10 or more dark green patches where the log polar model classifies 10+% addiitonal images correctly compared to the Euclidean model. BYOL never has a large dark green chunk of that size, and instead only gets 2 or 3 of those large performance gap pairings. This follows the trend we observed during training, where BYOL models were more sensitive to hyperparameter changes. BYOL was also the only model of the three primary models that collapsed during training. We believe this has to do with the model's lack of explicit solution to a mode collapse

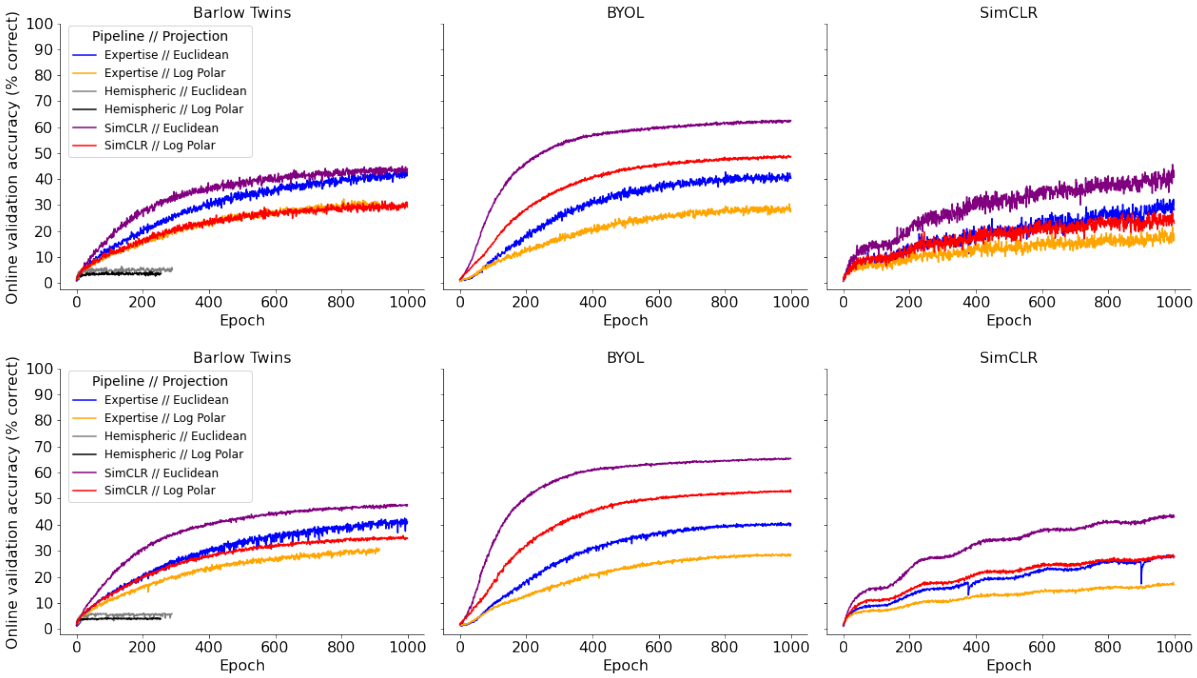

Figure 5.2. Dogs dataset training plots - online evaluator accuracy on training set (top row) and validation set (bottom row), various models

issue, instead opting for an exponential moving average teacher network that serves as a heuristic solution that experimentally keeps training stable. The Barlow Twins model lies somewhere in the middle of BYOL and SimCLR. It has sporadic dark green patches and does not have as many rotation-scaling pairs where log polar outperforms Euclidean compared to SimCLR. However, the Barlow Twins model loses less accuracy in upright, original size image performance in 5 of the 6 dataset-pipeline pairs tested. When applied to the faces dataset with the Expertise pipeline, the log polar model consistently outperforms the Euclidean model at all scales and rotations, including at original size upright images. This dataset-model-pipeline triplet, as well as BYOL on the faces dataset paired with the SimCLR pipeline, are the only two instances where we see log polar models having a higher test set accuracy than Euclidean models.

Comparing between pipelines is fairly difficult. There is no one pipeline that strictly generalizes better than the other options. However, we can examine some general trends. Barlow Twins appears to have an affinity for the Expertise pipeline, showing large amounts of dark green

cells on the faces dataset and Imagenet100 dataset when paired with Expertise pipeline compared to SimCLR pipeline. The BYOL model performs better with the SimCLR pipeline, although there aren't many high-performing rotation-scale combinations with either pairing.

5.3 Linear Regression Coefficient Analysis

We fit ordinary least squares linear regression models to predict classification accuracy, given dataset, model, transformation pipeline, projection type. We analyze the coefficients and standard error of the resulting model to determine and quantify any statistically significant performance gaps. We start with a simple, main effects model that attempts to model the effects of dataset choice, model choice, transformation pipeline, and projection (Euclidean or log polar) on final accuracy. The details of the regression fit are provided in Table 5.1, and the coefficients are provided in Table 5.2. This model is fit on upright, original size image performance only.

Table 5.1. Summary of simple linear regression model to predict classification accuracy

Model:	OLS
Formula:	`acc` $\sim$ `dataset + model + pipeline + projection`
R-squared:	0.926
Adj. R-squared:	0.906

This fit shows an R-squared of 0.926, and an adjusted R-squared of 0.905. The intercept term indicates the predicted performance of a Barlow Twins model fit on the Imagenet dogs dataset with the Expertise pipeline in a Euclidean projection. The model predicts a 60.5% test accuracy for such a model. Fitting on the faces dataset has a statistically significant higher accuracy prediction - we can see in Figure 3.1 that images are relatively similar in pose and image composition. The p-value for the coefficients for BYOL and SimCLR are higher than a significance level of 0.05 - statistically, both BYOL and SimCLR should have similar accuracy to Barlow Twins. SimSiam, which failed to fit properly as seen in Figure 5.3, has a statistically significant difference in accuracy from the other models. The hemisphere and modified SimCLR pipelines have a lower accuracy than the Expertise pipeline; the coefficient for the original

Table 5.2. Coefficients of simple linear regression model to predict classification accuracy

	Coef.	Std.Err.	$P > \|t\|$
Intercept	0.6053	0.0302	0.0000
dataset[T.faces]	0.1470	0.0276	0.0000
dataset[T.imagenet100]	0.0512	0.0272	0.0671
model[T.byol]	0.0008	0.0282	0.9776
model[T.simclr]	-0.0305	0.0270	0.2654
model[T.simsiam]	-0.5237	0.0551	0.0000
model[T.supervised_siamese_net]	-0.0721	0.0551	0.1976
pipeline[T.hemisphere]	-0.4556	0.0350	0.0000
pipeline[T.newsimclr]	-0.4142	0.0493	0.0000
pipeline[T.simclr]	0.0409	0.0243	0.1006
projection[T.lp]	-0.0648	0.0213	0.0042
projection[T.polar]	-0.2409	0.0640	0.0005

SimCLR pipeline has a p-value over 0.05 indicating no effect on accuracy. Switching to a log polar projection should reduce accuracy, with a statistically significant negative coefficient. Generally speaking, these results line up with what we've seen already in prior discussions on results.

Next, we look at how these models do at different levels of rotation and scaling. We build two linear regressions, one that includes projection and one that includes transformation pipeline. Both linear regressions include interaction effects so we can see if changing the pipeline or projection will lead to a difference in the model's generalizability. We don't include dataset or model type in these regressions to mitigate the chance that one dataset-model pair happened to perform better by random chance. This allows us to more closely investigate the impact of rotation and scaling overall. The results of these models are included in Tables 5.3, 5.4, 5.5, and 5.6.

Neither model has a particularly strong fit, with R-squared values around 0.44. However, we can still gain valuable insights. When we look at the rotation and scaling Pipeline regression (Tables 5.3 and 5.4), we can see that the coefficient for both padding and rotation are negative. Rotating the image away from the upright position or adding padding around the image to scale

Table 5.3. Summary of "rotation and scaling" Pipeline linear regression model to predict classification accuracy. $*$ indicates an interaction effect

Model:	OLS
Formula:	`acc ~ pipeline * padding + pipeline * rotation`
R-squared:	0.420
Adj. R-squared:	0.417

Table 5.4. Coefficients of "rotation and scaling" Pipeline linear regression model to predict classification accuracy

	Coef.	Std.Err.	$P > \|t\|$
Intercept	0.4143	0.0137	0.0000
pipeline[T.simclr]	-0.0462	0.0185	0.0126
padding	-0.0060	0.0004	0.0000
pipeline[T.simclr]:padding	0.0014	0.0005	0.0053
rotation	-0.0011	0.0001	0.0000
pipeline[T.simclr]:rotation	-0.0001	0.0001	0.5010

it down will both lead to a performance decrease. However, the SimCLR pipeline has a positive interaction coefficient with padding. When this is considered, at very small scales, models trained with the SimCLR pipeline should outperform models trained with the Expertise pipeline.

When we look at the rotation and scaling Projection regression (Tables 5.5 and 5.6), we can see that the coefficient for both padding and rotation are negative. Rotating the image away from the upright position or adding padding around the image to scale it down will both lead to a performance decrease. The log polar projection does not have a statistically significant interaction coefficient with either padding or rotation. Models trained with either projection should have roughly equivalent test set accuracy.

Finally, we look at the interaction effect of model type and the pipeline and any relationship that has to test accuracy. We fit a linear regression model to predict upright original size test accuracy given the combination of model type and pipeline, modeled as an interaction effects model.

The fit of this model is quite poor, with an R-squared value of 0.254. The interaction

Table 5.5. Summary of "rotation and scaling" Projection linear regression model to predict classification accuracy. $*$ indicates an interaction effect

Model:	OLS
Formula:	`acc ~ projection * padding + projection * rotation`
R-squared:	0.442
Adj. R-squared:	0.440

Table 5.6. Coefficients of "rotation and scaling" Projection linear regression model to predict classification accuracy

	Coef.	Std.Err.	$P > \|t\|$
Intercept	0.3639	0.0127	0.0000
projection[T.lp]	0.0501	0.0180	0.0056
padding	-0.0051	0.0004	0.0000
projection[T.lp]:padding	-0.0001	0.0005	0.7877
rotation	-0.0012	0.0001	0.0000
projection[T.lp]:rotation	0.0001	0.0001	0.2790

of model and transformation together cannot accurately predict test accuracy. If we look at the coefficients, none of the estimates are statistically significant. This matches the results we see in Chapter 4 and in Section 5.2 where no model consistently outperforms all competitors on all datasets. There may be some confounding factor that can better explain the difference in performance.

Table 5.7. Summary of interaction effects model to predict classification accuracy

Model:	OLS
Formula:	`acc ~ model * transformation`
R-squared:	0.254
Adj. R-squared:	0.129

Table 5.8. Coefficients of interaction effects model to predict classification accuracy

	Coef.	Std.Err.	$P > \|t\|$
Intercept	0.5224	0.0451	0.0000
model[T.byol]	0.0821	0.0638	0.2075
model[T.simclr]	0.0350	0.0638	0.5868
transformation[T.simclr]	0.0338	0.0638	0.6001
model[T.byol]:transformation[T.simclr]	0.0352	0.0902	0.6991
model[T.simclr]:transformation[T.simclr]	0.0834	0.0902	0.3622

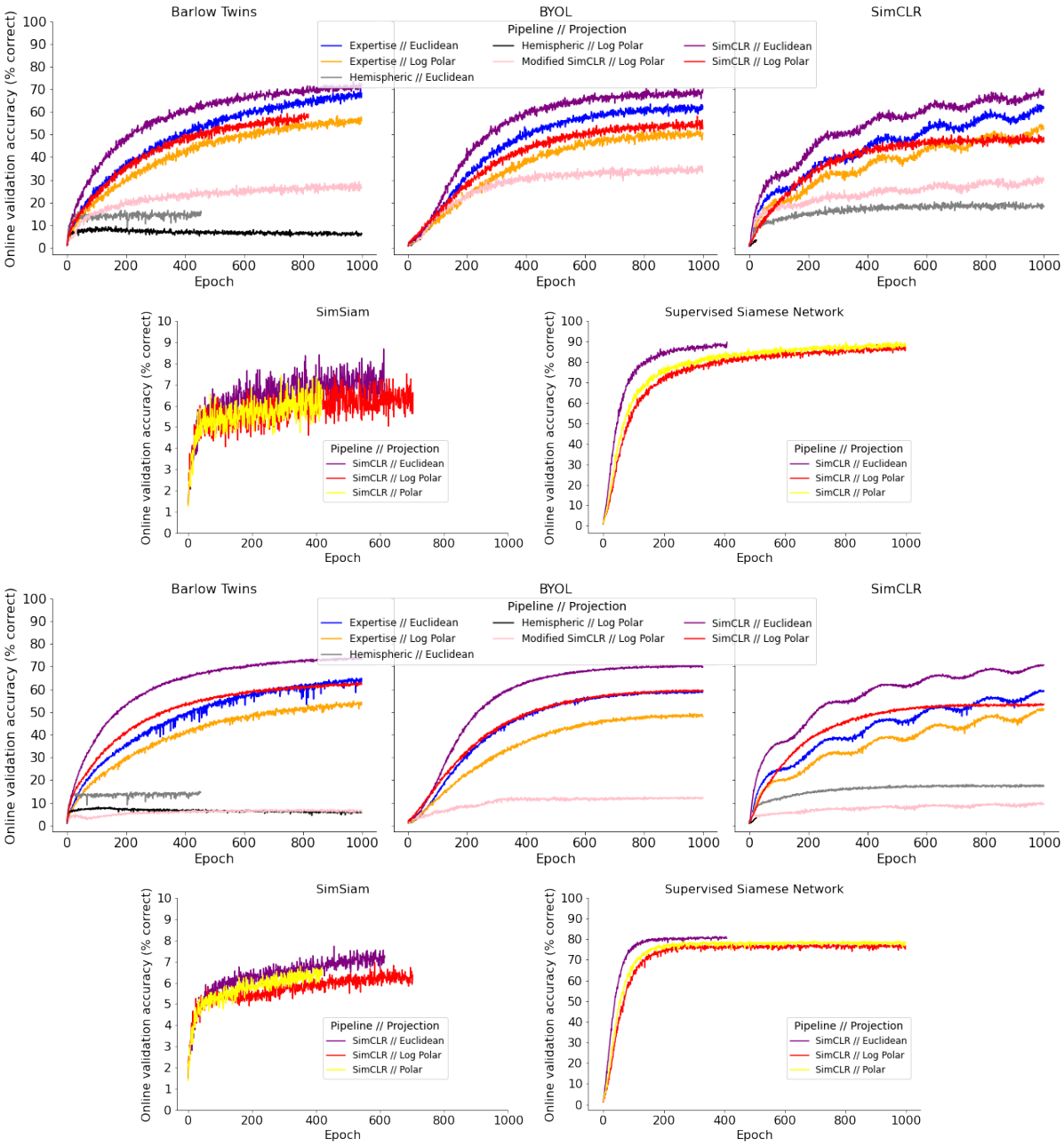

Figure 5.3. Imagenet 100 dataset training plots - online evaluator accuracy on training set (top two rows) and validation set (bottom two rows), various models

Figure 5.4. Heatmap of performance gap between log polar and Euclidean models, various rotations and scales, Imagenet 100 dataset

Figure 5.5. Heatmap of performance gap between log polar and Euclidean models, various rotations and scales, Imagenet Dogs dataset

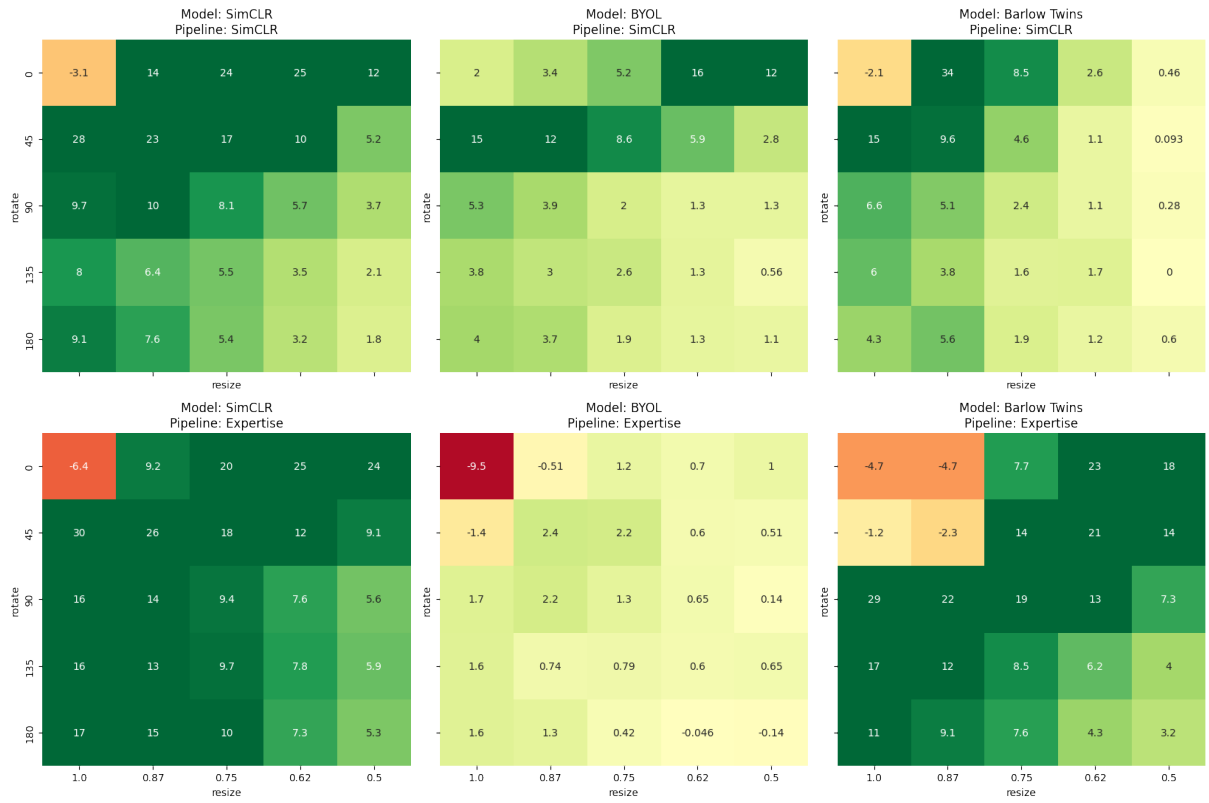

Figure 5.6. Heatmap of performance gap between log polar and Euclidean models, various rotations and scales, Faces dataset

Chapter 6

Conclusion

We test the use of the log polar projection in self-supervised learning models. We focus on multiple state of the art models - SimCLR, BYOL, SimSiam, Barlow Twins, and a supervised Siamese network baseline. The image transformation pipeline plays a large role in self-supervised learning. We use the SimCLR pipeline and Expertise pipeline from prior works, as well as a hemispheric pipeline that models the human visual system's hemifields and a modified SimCLR pipeline designed for log polar space. These model-pipeline pairs are each evaluated on the faces dataset, Imagenet dogs dataset, and Imagenet100 dataset. We use both subordinate level classification with the faces and Imagenet dogs dataset and basic level classification with the Imagenet100 dataset. We evaluate our models at multiple rotations and scales to better understand how model performance is impacted by transformations to inputs at test time.

Models trained with the log polar projection have a lower test set accuracy compared to models trained with the Euclidean projection. However, log polar models show better generalization than Euclidean models, especially on out-of-distribution images. Log polar models have a higher test set accuracy at multiple rotations and scales. This is due to the partial invariance to scale and rotation granted by the log polar transform. In most cases, the model loses some small amount of accuracy on original size upright images, but shows a significant improvement on transformed images. This improvement on out-of-distribution images makes

the log polar transform a compelling option for certain classes of applications that require the ability to generalize to out-of-distribution images. While this is not a thorough testing of the model's sensitivity to distribution shift, we do believe this is a promising sign for the usability of the log polar transform.

Due to the resolution of source images, log polar transformed images showed artifacts. Future work can explore the use of a higher resolution dataset, paired with a tuned modified SimCLR pipeline for a higher quality log polar image. Further, the hemispheric pipeline can be tuned to include overlapping patches between the two transformed images, similar to the binocular portion of the human visual field. Finally, curriculum learning where a model is gradually introduced to new classes has been shown to help with difficult tasks. Integrating a curriculum learning approach to log polar models would better model the human developmental process of gradual exposure to new ideas.