Dog breeds classification with Siamese Network

Jianing Zhang, Renjie Ni, Yi Lin[1]

Abstract

Dogs are one of the most beloved domestic animals, with over 300 different breeds worldwide and Dog breed classification is one of the most important task in the field of computer vision and animal welfare. However, identifying a dog's breed is a challenging problem for the computer vision community. We did not find academic papers based on dog breed classification, and online communities like Kaggle did not achieve a satisfying performance ($< 80\%$ accuracy). In this project, we propose to use Siamese Neural Network (SNN) to complete this task. Compared to traditional convolution networks (CNN), SNN learns the dissimilarity measure between pairs of images, and has superior performance in applications such as face recognition and signature verification. In addition, we used prototypes to construct dissimilarity space which is fed into a traditional classifier to predict the breed. We believe SNN's success in recognition task will help this system to learn the dissimilarties of dog breeds, and achieve a potentially better performance than traditional CNN. Indeed, we have achieved an accuracy with SNN that is very close to CNN method.

1. Introduction

Dogs have been domesticated for thousands of years and have become one of the most beloved and popular pets worldwide. With over 300 different breeds recognized by the American Kennel Club (AKC) and many more breeds worldwide, dogs come in a wide variety of shapes, sizes, colors, and temperaments. Identifying a dog's breed is not only important for breeders and owners, but also for veterinarians, animal welfare organizations, and researchers.However, identifying a dog's breed can be challenging, especially for those who are not familiar with dogs. Physical characteristics alone may not be sufficient to identify a dog's breed accurately, as some breeds may have similar physical features, while others may have unique genetic traits that are not visible to the naked eye. Incorrect breed identification can lead to inappropriate care, training, and handling, resulting in serious consequences for the dog's health and behavior.

While previous work on dog breed classification has primarily focused on the use of convolutional neural networks like ResNet, VGG, and Inception, these networks often require a large number of images to achieve high performance. This can be a challenge in some cases, as obtaining a sufficient number of images for certain dog breeds can be difficult or time-consuming. However, recent advancements in neural network architectures have led to the development of the Siamese Neural Network (SNN) (Wu et al., 2017) (Koch et al., 2015), which offers a promising solution for this issue. SNNs are a type of neural network that can calculate the similarity between two images, making them well-suited for tasks like dog breed classification where there may be a limited number of images available for certain breeds. One of the main advantages of SNNs is their ability to learn from few examples (i.e. few-shot learning), making them a valuable tool for situations where obtaining a large number of images is not feasible. Additionally, SNNs have been shown to perform well in a variety of image-related tasks, including face recognition, object tracking, and image retrieval.

In order to enhance the performance and robustness of our dog breed classification system using the Stanford dog data set, we construct a system that utilize a combination of Siamese networks, dissimilarity space with prototypes, and 1-vs-all Support Vector Machine (SVM). Additionally, we also attempted to use SIFT and LBP descriptors, as well as ensemble learning with other CNN models.

SIFT (Scale Invariant Feature Transform) (Lowe, 1999) is a widely-used feature extraction technique that can identify and describe the local features of an image, making it well-suited for image recognition tasks such as dog breed classification. LBP (Local Binary Pattern) (Hadid, 2008) is another feature extraction technique that describes the texture of an image by encoding local patterns of pixel intensities. By combining these two feature extraction techniques with a

[1]University of Waterloo, Waterloo, Ontario, Canada. Correspondence to: Jianing Zhang <j249zhan@uwaterloo.ca >, Renjie Ni Zhang <r6ni@uwaterloo.ca >, Yi Lin <y477lin@uwaterloo.ca >.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Siamese neural network architecture, we aim to create a more robust and accurate model for dog breed classification that can better handle variations in image quality, lighting, and other factors that may affect image recognition.

The rest of the report is organized as follows. Section 2 listed some related works. We introduce Siamese Neural Network separately in Section 3. Section 4 takes some time to discuss the main methodology in this project, where the main classification system is explained. Section 5 listed some experiments we did, and the results are shown and discussed in Section 6.

2. Related Works

2.1. Others' Work

While several studies have been conducted on image classification of celebrities and animal types, there has been a lack of research on classifying dog breeds. Identifying dog breeds is challenging, as breeds cannot be determined by appearance alone. Online communities like Kaggle have not achieved satisfactory accuracy in dog breed classification using traditional CNN models such as AlexNet (Krizhevsky & Hinton), VGG-16 Net (Simonyan & Zisserman), ResNet (He & Sun, 2015) on the Stanford dog dataset, containing 20,580 images of 120 different breeds worldwide.

2.2. Siamese Neural Network

This project is inspired by the Siamese Networks (Wu et al., 2017). Siamese Neural Network (Nandy et al., 2020), or SNNs, is one of the most popular neural network architectures that use One-Shot Learning (Koch et al., 2015) and Few-Shot Learning (Müller et al., 2022), and can predict multiple classes from very little data. This ability has made Siamese neural networks very popular in real-world applications in security, face recognition, signature verification, and more.

2.3. SIFT and LBP

SIFT (Lowe, 1999) and LBP (Hadid, 2008) features can be beneficial for capturing and describing the local features and texture patterns of an image, which can be critical for accurate classification. Using a combination of SIFT and LBP features can help to enhance the robustness and accuracy of the classification system. SIFT (Lowe, 1999) features are invariant to scale, orientation, and affine distortion, which means they can still be accurately matched even if the images of the same dog breed are taken from different angles, distances or lighting conditions. LBP (Hadid, 2008) features are robust to noise, and can accurately capture the texture information of an image, making them well-suited to handle variations in image quality or other factors that may affect image recognition.

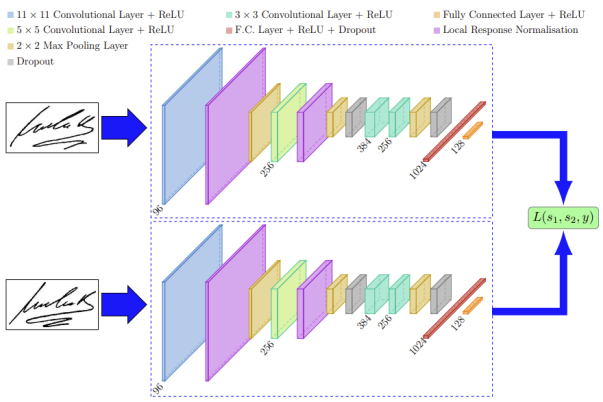

Figure 1. An example of SNN architecture for signature verification. 2 images of signiture are fed into a series of convolutional layer, then flattened and reduced to two embedding vectors of dimension 128, which are fed into loss function directly with the label.

3. Siamese Neural Network

A siamese neural network (SNN) is a class of neural network architectures that contain two or more identical subnetworks. Here, "identical" refers to same configuration with the same parameters and weights. Parameter updating is mirrored across both sub-networks and it's used to find similarities between inputs by comparing the two embedding vectors corresponding the two inputs. The exact replication of two neural-nets ensures that if fed with the same images, then the output embeddings of SNN is exactly the same. Fig 1 is an example of Siamese Neural Network architecture from (Dey et al., 2017) that is used for signature verification. Other applications of SNN includes face recognition.

Since the 2 neural networks are identical, SNN focuses on learning embeddings (in the deeper layer) that place the same classes / concepts close together. Hence, it is able to learn semantic similarity, which differs SNN from traditional neural network classification models. Given this, simply averaging it with a classifier can do much better than averaging two correlated supervised models. The greatest advantage of an SNN is its ability to achieve a better recognition result using fewer number of samples, even imbalanced dataset. Therefore SNN is particularly well-known for one-shot learning (Koch et al., 2015). On the other hand, the drawback is obvious. Since an SNN involves learning from quadratic pairs (to see all information available), they're less efficient, in terms of speed, than the normal classification type of learning.

In our specific cases, the input of SNN are pairs of dog images and a binary label, representing whether the two images come from the same object (recognition task) / class (classification task). SNN uses 2 identical convolutional neural networks (CNN) to process the image, then uses a feed forward dense network to obtain two embedding vectors. Specific architectures that we used will be explained in the Section 5: Experiment .

4. Methodology

In this section, we will introduce the stages of our project. We will start with obtaining and engineering the Standford Dog Dataset, followed by defining the general architecture that we inferred from literature review. Finally, loss function used for training along with other evaluation metrics will be presented. We will discuss each component in detail, and share our thoughts during the project.

4.1. Dataset and Preprocessing

4.1.1. About the Dataset

In this project, we used stanford dog dataset. The Stanford Dogs Dataset contains images of 120 breeds of dogs from around the world. This dataset has been built using images and annotation from ImageNet for the task of fine-grained image categorization. It was originally collected for fine-grain image categorization, a challenging problem as certain dog breeds have near identical features or differ in colour and age. There are 20,580 images in total, out of which 12,000 are used for training (100 images for each breed) and 8580 for testing. Class labels and bounding box annotations are provided for all the 12,000 images.

4.1.2. Data Augmentation

For each image in the dataset, it has 3 RGB channels with 224 * 224 pixel for each channel, and a corresponding breed label. Before starting to build our model, we did some data augmentations. In the first, we did a resize step, resizing the original size to 100 * 100 to reduce the size of the image. Beyond that, we also tried to rotate the images by a small angle in both clockwise and counter clockwise, change the brightness of the images, flip the images vertically and horizontally and add noises which follow different distributions including Gaussian distribution. However, after building the model and testing them on the test set, we found that the most useful part is resizing. In this case, we only kept the resizing code in the submission. We also applied SIFT, since using SIFT to detect and describe the key features of the original images, we can create more diverse and realistic augmented images that accurately capture the variations and distortions that occur in real-world images.

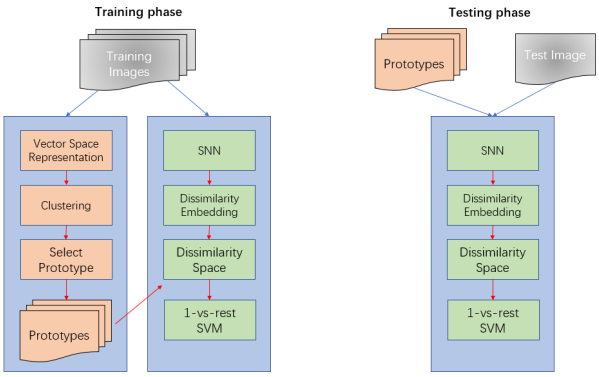

Figure 2. A basic outline of the proposed approach. In the training phase, prototype selection is performed, and an SNN is trained to define a dissimilarity measure; in the testing phase, each unknown pattern is represented by its distances to the prototypes and classified accordingly.

4.1.3. Label

After that, the processed images are made into pairs, since the input of SNN requires two images. The labels are defined as 0 and 1, where 1 represents the two images coming from the same class, and 0 represents the two images coming from different classes. Formally, a single element from the Siamese dataset is defined as: (image1, image2, label).

4.2. Purposed system

Siamese Networks are initially designed to learn similarity / dissimilarity of the two inputs. Most former works of SNN focuses on one-shot image recognition tasks (Koch et al., 2015), such as face recognition (Song et al., 2019), (Wu et al., 2017) and signature verification (Dey et al., 2017). Few works used SNNs for image classification tasks, however, and we would develop a system that uses SNN to classify dog breeds. We believe the success of SNN in measuring dissimilarity can be effectively applied on dog breeds as well.

Inspired by the work of Spectrogram Classification Using Dissimilarity Space (Nanni et al., 2020), the proposed method for dog breed classification using dissimilarity space is based on several steps which are schematized in Fig 2. In order to define a similarity space, it is necessary to select a distance (a.k.a dissimilarity) measure and a set of prototypes in the training phase.

An SNN is trained to maximize the dissimilarity between couples of dogs of different breeds, while minimizing the dissimilarity for couples of the same breed. The output of the SNN is two embedding vectors that represent the input images x and y. The set of prototypes $P = \{p_i | i \in [K]\}$

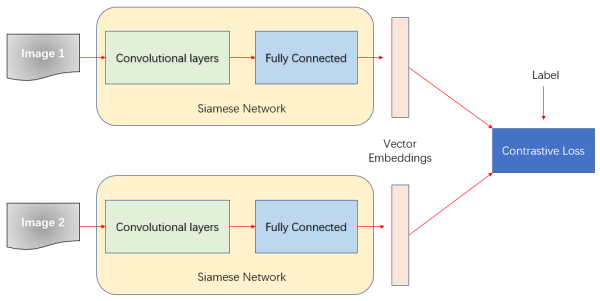

Figure 3. A basic structure for Siamese Network Training. An SNN consists of a sequence of convolutional layers, a sequence of fully connected layers. Note that the top and bottom SNN are exactly the same. They share the same weights and optimizer. The output vector embeddings and the label is used to calculate contrastive loss.

are selected as the k-centroids of the clusters generated by a supervised clustering algorithm. The final step represents each training sample x in the dissimilarity space by a feature vector $f \in \mathbb{R}^k$, where each component f_i is the distance between x and the prototype p_i: $f_i = d(x, p_i)$, which is calculated by the Euclidean distance of the embedding vectors. These feature vectors are used to train an SVM for the final classification task. In the testing phase, each unlabeled dog is first represented in the dissimilarity space by calculating its distance to all the prototypes, then the resulting feature vector is classified by SVM.

4.2.1. Siamese Neural Network Training

The SNN used in this system has 2 identical neural-networks that is consists of two parts: a sequence of dilated convolutional layers and a sequence of linear dense layers. The outputs are two embedding vectors that represents the input images. With the help of true label, the SNN is trained with ADAM optimizer with minimizing contrastive loss function. Also defined earlier, label 1 represents the two images coming from the same class, and 0 represents the two images coming from different classes. Fig 3 shows a graph visualization of this structure.

Contrastive Loss Function

Let x_1, x_2 be the two input images and $y \in \{0, 1\}$ be the corresponding label fed into the SNN. Let $\mathbf{v}_1, \mathbf{v}_2$ be the corresponding output embedding vector. Let $d(\mathbf{v}_1, \mathbf{v}_2) = \|\mathbf{v}_1 - \mathbf{v}_2\|_2$. Define M as the "margin". Then the contrastive loss function $L(\mathbf{v}_1, \mathbf{v}_2, y)$ is defined as:

$$\begin{aligned} L(\mathbf{v}_1, \mathbf{v}_2, y) =& (y)\frac{1}{2} d(\mathbf{v}_1, \mathbf{v}_2)^2 \\ &+ (1-y)\frac{1}{2}\left\{\max\left(0, M - d(\mathbf{v}_1, \mathbf{v}_2)\right)\right\}^2 \end{aligned} \tag{1}$$

And the total loss is the sum over all possible image pairs.

When two dogs come from the same class, $y = 1$, and L only takes value of the first term, $\frac{1}{2} d(\mathbf{v}_1, \mathbf{v}_2, y)^2$, which is simply the squared distance. i.e. The model minimizes the Euclidean distance between the two embedding vectors when dogs are the same breed. When two dogs come from different breeds, $y = 0$ and L takes value of the second term, which is the Hinge loss. The model separate the embedding vectors further and further away until we hit the margin M. This is the idea loss function to use for dog classification task: when two dogs are the same breed, we want to minimize their dissimilarity, and vice versa.

4.2.2. Prototype Selection

In this phase, k prototypes are extracted from the training set. In theory, every dog image in the training set could be selected as a prototype, but this would be too computationally expensive and the generated dissimilarity vectors would be too large resulting in the curse of dimension. An alternative way is to employ clustering techniques to compute k-centroids for each class. Clustering would significantly reduce the dimension of the resulting dissimilarity space and thus make the process more practical.

On the other hand, the k-centroids would be dog "representative" of each breed. Unlike human faces, dogs from the same breed could have different appearances. For example, the famous golden retriever could be either gold, white or multi-colored. It is important to have $k > 1$ in this case.

For vector representation of the images, LBP method is used. For simplicity, we used supervised k-means clustering method to obtain the 5-centroids for each breed using the vector representation of images.

4.2.3. Dissimilarity Definition

After training SNN and selecting prototypes, we use these result to construct a dissimilarity space. Specifically, for each image x (training / testing set depending on train / test stage), let $f(x) \in \mathbb{R}^K$ be the dissimilarity feature vector, where K is the total number of prototypes.

$$f(x) = [f_i]_1^K = [d(SNN(x), SNN(p_i)) \quad \forall i \in [K]]$$

where d is the Euclidean distance. Stacking $f(x)$ vertically by image x, we obtain the dissimilarity space of the whole

dataset $F = \mathbb{R}^{N \times K}$ where N is the total number of images. F is the input of SVM Classifier in the next stage.

4.2.4. SVM CLASSIFIER

Support Vector Machine (SVM) is a binary classifier that searches for a hyperplane that separates data. Prediction is a matter of mapping an unseen pattern to the side of the hyperplane that represents its class. If the data are not linearly separable, kernel functions can be employed to map the data into higher-dimensional spaces where the data can be separated. Although SVM is a binary classifier, it can also handle problems by training an ensemble of SVMs and then by combining their decisions using a one-against-all method that classifies a pattern as belonging to the class with the highest confidence score. Such approach is taken in this project.

4.3. Evaluation

For this project, we use the contrastive loss function to evaluate the performance of SNN. In previous sections, we have discussed the concept of "margin" M. We observe that M is the smallest value of dissimilarity measure when two dogs are not from the same breed in the training phase. Thus the following evaluation method is designed: with a fixed random seed, we will pick a fixed number of image pairs and corresponding label(x_1, x_2, y) randomly from the couple dataset. Then feed them through the trained SNN and obtained two embedding vectors $\mathbf{v_1}, \mathbf{v_2}$. We will calculate their Euclidean distance (i.e. dissimilarity) and set M as the threshold of label prediction. If the dissimilarity is greater than M, the prediction is that the two images comes from different classes so $\hat{y} = 0$, and vice versa. Compare y and y_hat for all randomly chosen image pairs and obtain an accuracy measure. We will call this **"SNN accuracy"**.

To evaluate the whole system, we will simply calculate the accuracy based on the output of one-against-all SVM classifier for each dog, and compare its results to the ground truth. We will call this **"Overall Accuracy"**.

5. Experiments

5.1. Limitation in Computation Resource

First of all, we would like to explain in advance a restriction that we encounter. As mentioned earlier, one of the disadvantage of SNN is its surprisingly long training time. Stanford Dog Dataset has 120 breeds and over 12,000 images in total. Training an SNN that consists of convolutional neural nets is extremely resource expensive. We did a pilot run using Colab Pro, and a single model training for all 120 breeds took over 12 hours. This is expected since each breed has 100 images, and the number of pairs grows exponentially with the number of class. Due to this cruel reality, we have to buy some computation resource (Colab Pro), and decided to reduce to only 5 breeds while keeping all the images in each breed for this project. Even with Colab Pro, the computation unit is not unlimited. Therefore the experiments performed are carefully selected.

5.2. Baseline: Random Guess and ResNet-16

Siamese Network is the main character in this project, yet we still want to compare its performance on the classification task with traditional Convolutional Neural Network. We select ResNet-16 here.

5.3. SNN Training

SNN training consists of 2 parts. We mainly focuses on exploring different architectures of SNN. In Appendix B, we have listed the architectures that we purposed for the dog breed classification tasks in two tables. Note that SNN1, SNN2 and SNN5 is capable of dealing with images of size 100×100, while SNN3 and 4 deal with size 224×224. Within the same image size, we chosed to compare the performance under embedding vectors of low and high dimension. For example, for size 100×100, SNN1 outputs embedding vectors of dimension 2 while SNN2 outputs 1024. Also notice that we tried out number of architectures that is not listed in these tables. We only presented the architecutres whose results are worth discussing.

On the other hand, we also tuned the hyperparameters and use regularization techniques to avoid overfitting. Hyperparmaters include number of epochs, on which the model is trained until convergence; batch size; and learning rate for ADAM optimizer. We also used the following regularization techniques: batch normalization, which is placed after each 2D convolutional layer; early stopping; using L_2 regularization on the contrastive loss function.

5.4. SVM tuning

We use grid search to optimize the performance of SVM. This is a very simple task and requires less resource than SNN. Hyperparamters include penalty term C and kernel function.

6. Result

Let me start with the tuning result of SVM, as this is the simplest part of the system. Hyperparameters are quite deterministic here, since tuning SVM does not really depend on the performance of SNN, and is done separately. The best performance of SVM uses polynomial kernel functions and sets $C = 1$.

The main result is shown in the table in Appendix A (Apologize for the inconvenience since the table is too wide). For a

baseline reference, random guess achieves 11.21% accuracy, and a ResNet-16 without tuning achieved 76.34%.

For every selected SNN models in this report, we tuned the hyperparameters mentioned in Section 5.3, and obtained a "best" performance for each of them. We trained all models until 500 epochs if not converged. The definition of "SNN accuracy" and "Classification accuracy" can be found in Section4.3. We make the following observations:

- **ResNet-16 is indeed one of the best CNN models.** None of our proposed architectures (SNN 1-4) has defeated ResNet-16. So we had an idea of directly replacing the convolutional layers with a pretrained ResNet-16 model (see SNN5). This boosted the testing accuracy by 22%! This is best model from this project.
- **It is important to have a classifier after SNN.** We can treat "SNN accuracy" equivalent to classifying pair of images using only the margin (from contrasitve loss) as a threshold on dissimilarity measure. The classification accuracy with SVM is, in general, higher than the "SNN accuracy". This means dissimilarity measure alone is not a good criterion for classification in this dataset. Previous works used dissimilarity measure to determine if 2 faces belong to the same person. However even if two dogs are the same breed, they do not necessary to have the same physical appearance. Therefore we need more a entire dissimilarity space with prototypes as cross-reference. This result is very reasonable.
- **Smaller size images tend to give better result in this dataset.** We found SNN1 an 2 perform better than SNN3 and 4 on all accuracies. This result is not generalizable. It could be adjusted with the architecture of CNN layers and FC layers, yet we didn't have the time to verify it. On the other hand, SNN3 and 4 did not converge within 500 epochs. So another explanation is that these models are under-fitting.
- **Larger embedding vectors tend to do better.** If we compare SNN 1 with 2, and compare 3 with 4, we would realize that outputting higher dimensional embedding vector slightly improves the accuracies. One possible explanation is that we need enough number of information to determine dog breeds. We would also like to try other embedding dimension if possible in the future.
- **SVM is very accurate on the training set.** We achieved 100% training accuracy with SVM on images of size 100×100. This is a shocking result, meaning that by comparing a seen dog with all 5-centroids prototypes, we can perfectly determine its breed using the system. This is a positive feedback for our system on one hand, but also a mark of overfitting on the other hand.
- **All of the models are overfitting.** This is very direct result from the table, and the models we are using are already been dealt with overfitting problem. For example, we use Batch normalization, L_2 regularization on the loss function, data augmentation and so on. Dropout was not helping to reduce the gap between train and test accuracy, yet still lowering the overall accuracies.

One of the most important thing that we could extend from this project is to engineer the data more carefully. Note that in this dataset, there are 100 training images for and around 80 testing images each breed . This means the testing set size is 80% of the training set. After the augmentation done already, the proportion became around 40% which is still too high. We need to find other methods to increase the training size so that the model can generalize better to the testing size.

Our best model didn't beat ResNet-16, yet its performance is very close to a traditional CNN classifier. Although previous works has shown SNN's success in image recognition, we still believe with further resource (time & computational resource), Siamese Neural Network is able to handle the classification of dog breeds.

7. Conclusion

Our project focuses using Siamese Neural Network to classify dog breeds using Standford Dog Dataset. We purposed a classification system that utilize Siamese Neural Network, dissimilarity space, and Support Vector Machine to work on the dog breed dataset. This system has achieved very close performance to the traditional CNN classifier. Our success includes using pretrained ResNet as part of the SNN architecture; selecting prototypes to lower the dimension of dissimilarity space; and using a one-against-all SVM classifier on top of the dissimilarity measure. Due to limited time and computational resources, we did not fully explore potentially better architectures and other data preprocessing methods. We believe with proper handling of the original data and more tuning on the model, it can achieve a better performance. Still, our project is a good attempt to use SNN and dissimilarity measure to classify dog breed, a challenging and diverse dataset.

References

Dey, S., Dutta, A., Toledo, J. I., Ghosh, S. K., Lladós, J., and Pal, U. Signet: Convolutional siamese network for writer independent offline signature verification. *CoRR*,

abs/1707.02131, 2017. URL http://arxiv.org/abs/1707.02131.

Hadid, A. The local binary pattern approach and its applications to face analysis. pp. 1–9, 2008. doi: 10.1109/IPTA.2008.4743795.

He, K., Z. X. R. S. and Sun, J. Deep residual learn- ing for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Koch, G., Zemel, R., Salakhutdinov, R., et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

Krizhevsky, A., S. I. and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM, 60(6):84–90, 2017.*

Lowe, D. G. Object recognition from local scale-invariant features. 2:1150–1157, 1999.

Müller, T., Pérez-Torró, G., and Franco-Salvador, M. Few-shot learning with siamese networks and label tuning, 2022.

Nandy, A., Haldar, S., Banerjee, S., and Mitra, S. A survey on applications of siamese neural networks in computer vision. pp. 1–5, 06 2020. doi: 10.1109/INCET49848.2020.9153977.

Nanni, L., Rigo, A., Lumini, A., and Brahnam, S. Spectrogram classification using dissimilarity space. *Applied Sciences*, 10(12):4176, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556, 2014.*

Song, L., Gong, D., Li, Z., Liu, C., and Liu, W. Occlusion robust face recognition based on mask learning with pairwise differential siamese network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 773–782, 2019.

Wu, H., Xu, Z., Zhang, J., Yan, W., and Ma, X. Face recognition based on convolution siamese networks. In *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pp. 1–5. IEEE, 2017.

A. Experiment Results

Baseline Model	**Test Accuracy**			
Random Guess	11.21			
ResNet	**76.34**			
SNN Models	**SNN Accuracy**		**Classification Accuracy with SVM**	
	Train Accuracy	Test Accuracy	Train Accuracy	Test Accuracy
SNN1	82.31	34.06	**100**	50.84
SNN2	83.48	40.91	**100**	52.08
SNN3	64.22	20.16	80.13	35.48
SNN4	67.68	26.72	84.65	39.92
SNN5	85.46	79.91	**100**	**72.91**

B. SNN architecture

Siamese NN 1				
Layers	Output Data Size	Kernel Size	Stride	Num.Filters
Input Layer	$3 \times 100 \times 100$			
2D Convolution	$96 \times 23 \times 23$	11×11	4	96
2D Batch Norm	$96 \times 23 \times 23$			
ReLU	$96 \times 23 \times 23$			
Max Pool	$96 \times 11 \times 11$	3×3		
2D Convolution	$256 \times 7 \times 7$	5×5	1	256
2D Batch Norm	$256 \times 7 \times 7$			
ReLU	$256 \times 7 \times 7$			
Max Pool	$256 \times 3 \times 3$	2×2		
2D Convolution	$384 \times 1 \times 1$	3×3	1	384
2D Batch Norm	$384 \times 1 \times 1$			
ReLU	$384 \times 1 \times 1$			
Fully Connected 1	1024			
Fully Connected 2	256			
Fully Connected 2	2			
Siamese NN 2				
Layers	Output Data Size	Kernel Size	Stride	Num.Filters
Input Layer	$3 \times 100 \times 100$			
2D Convolution	$96 \times 23 \times 23$	11×11	4	96
2D Batch Norm	$96 \times 23 \times 23$			
ReLU	$96 \times 23 \times 23$			
Max Pool	$96 \times 11 \times 11$	3×3		
2D Convolution	$256 \times 7 \times 7$	5×5	1	256
2D Batch Norm	$256 \times 7 \times 7$			
ReLU	$256 \times 7 \times 7$			
Max Pool	$256 \times 3 \times 3$	2×2		
2D Convolution	$384 \times 1 \times 1$	3×3	1	384
2D Batch Norm	$384 \times 1 \times 1$			
ReLU	$384 \times 1 \times 1$			
Fully Connected 1	1024			

Siamese NN 3				
Layers	Data Size	Filter Size	Stride	Num.Filters
Input Layer	$3 \times 224 \times 224$			
2D Convolution	$64 \times 215 \times 215$	10×10	1	64
2D Batch Norm	$64 \times 215 \times 215$			
Max Pool	$64 \times 107 \times 107$	2×2		
ReLU	$64 \times 107 \times 107$			
2D Convolution	$128 \times 26 \times 26$	7×7	4	128
2D Batch Norm	$128 \times 26 \times 26$			
ReLU	$128 \times 26 \times 26$			
2D Convolution	$128 \times 8 \times 8$	5×5	3	128
2D Batch Norm	$128 \times 8 \times 8$			
ReLU	$128 \times 8 \times 8$			
2D Convolution	$64 \times 5 \times 5$	4×4	1	64
2D Batch Norm	$64 \times 5 \times 5$			
ReLU	$64 \times 5 \times 5$			
Fully Connected 1	1024			
Fully Connected 2	256			
Fully Connected 3	2			
Siamese NN 4				
Layers	Data Size	Filter Size	Stride	Num.Filters
Input Layer	$3 \times 224 \times 224$			
2D Convolution	$64 \times 215 \times 215$	10×10	1	64
2D Batch Norm	$64 \times 215 \times 215$			
Max Pool	$64 \times 107 \times 107$	2×2		
ReLU	$64 \times 107 \times 107$			
2D Convolution	$128 \times 26 \times 26$	7×7	4	128
2D Batch Norm	$128 \times 26 \times 26$			
ReLU	$128 \times 26 \times 26$			
2D Convolution	$128 \times 8 \times 8$	5×5	3	128
2D Batch Norm	$128 \times 8 \times 8$			
ReLU	$128 \times 8 \times 8$			
2D Convolution	$64 \times 5 \times 5$	4×4	1	64
2D Batch Norm	$64 \times 5 \times 5$			
ReLU	$64 \times 5 \times 5$			
Fully Connected 1	1024			
Fully Connected 2	2048			
Fully Connected 3	4096			
Siamese NN 5				
Layers	Data Size	Filter Size	Stride	Num.Filters
Input Layer	$3 \times 100 \times 100$			
Pretrained ResNet	NA			
Fully Connected	1024			