Recognition and Representation of Facial Features using a Multi-Layer Capsule Network

Vikram Shenoy, Alexander Chowdhury, Harry Hartenstine
Khoury College of Computer and Information Science
Northeastern University
Boston, MA
shenoy.vi, chowdhury.al, hartenstine.h @husky.neu.edu

1. Objectives and Significance

Capsule Networks represent one of the newest breakthroughs in Computer Vision. A major disadvantage of the current state-of-the-art Convolutional Neural Networks is that their max pooling feature discards critical pieces of information. Capsule Networks retain this information by removing max-pooling and introducing "capsules", which encapsulate the pose of objects. These capsules are learned by using a novel routing algorithm called Dynamic routing. Sabour *et al*. [11] show that Capsule networks are able to yield better performance on the MNIST dataset than standard Convolutional Neural Networks, and similar performance as compared to deeper Convolutional Neural Network architectures. However, a standard capsule network is class-dependent in its secondary layer and will not be able to generate encodings for arbitrary input images.

In this paper, the original CapsNet architecture is revised to include multiple internal capsule layers. It is hypothesized that adding an extra layer will provide the required flexibility to generate an encoding that will be able to capture more complex relationships between different features. The proposed multi-layer capsule network should show that by using more capsule layers, the network will be able to generate better encodings through a deeper understanding of facial features as compared to the encodings generated by the original network.

2. Background

2.1. Convolutional Neural Networks

An early implementation, of the basic structure of modern Convolutional Neural Networks was the Neocognitron [1]. Drawing inspiration from elements of the visual cortex, it consists of alternating layers of "simple" S-cells and "complex" C-cells. Each S-cell in a layer responds to a different segment of an image

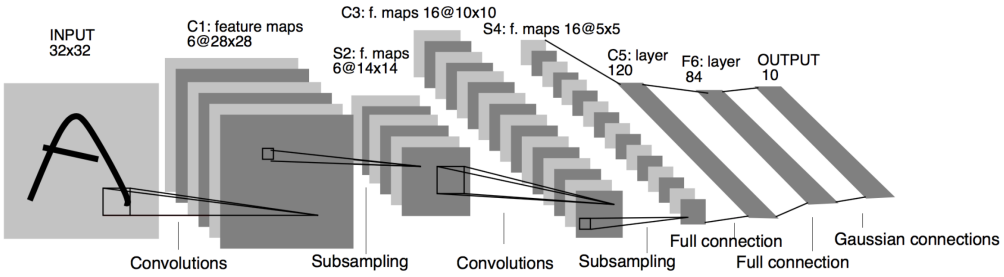

Figure 1. LeNet-5 from the original paper published by Y. LeCun *et al*. [8]

and weights are allowed to vary such that each cell extracts different features. Each C-cell has fixed and unmodifiable connections to groups of S-cells from the preceding layer. Layers of C-cells contribute a max-pooling effect which makes the network more resilient to changes in size or position of features and distortions. Subsequent layers thus find patterns among earlier detected features and reduce the total number of cells with each layer. The network is constructed with 7 layers such that the final layer has one cell activated for each class of input.

Lecun et. al. applied back-propagation to a smaller network [7]. The first of three hidden layers consisted of 12 8x8 feature maps, the second having 12 4x4 feature maps, and third being a fully connected layer connected to the output of 10 units. Rather than pooling layers, the first two layers had a set of 25 shared weights each to provide translation invariance. This small network required 3 days of training with resources available at the time. The latest version of this network, LeNet-5 [8] is shown in Figure 1.

Powerful graphics processing units fueled a great increase in scale of CNNs. AlexNet consisted of 5 convolutional and 3 fully connected layers [6]. New methods were required to prevent overfitting such as dropout and overlap max-pooling. Alexnet was trained for 5 to 6 days on two parallel GPUs. Following the success of Alexnet, the number of layers in high-performing image recognition networks has exploded. Microsoft implemented a deep CNN Resnet with 152 layers [4].

Despite the impressive error rates achievable with the latest CNNs, the networks remain vulnerable to certain pathologies including vulnerability to adversarial images [15]. Adversarial images seem virtually identical but have carefully crafted noise that cause networks to misclassify images with high confidence. In an extreme case, this can be done with one carefully chosen pixel [14]. Additionally, CNNs are translation invariant by design. This means that features re-arranged in a nonsensical way will be classified just the same as an unmodified image. Finally, the pooling strategies that prevent overfitting in deep CNNs also cause precise position information to be discarded. In applications requiring precise relationships between features, this is undesirable.

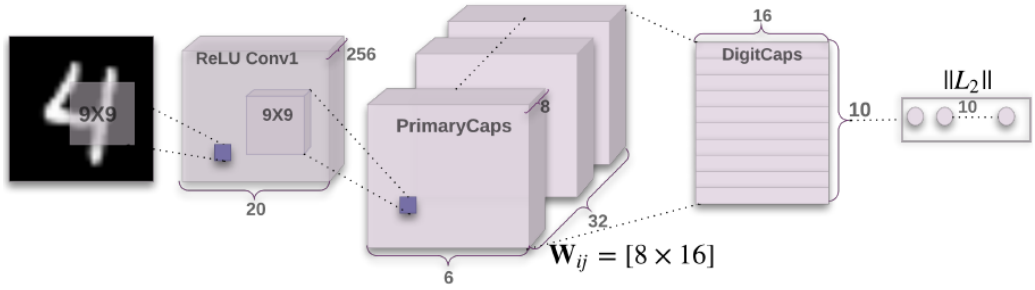

Figure 2. Encoder portion of Capsule Network by S.Sabour *et al*. [11]

2.2. Capsule Networks

Capsule networks overcome the limitations of CNNs by maintaining precise information about position and orientation of the image. Inspiration is drawn from computer graphics where a visual image is constructed from some internal hierarchical geometric data representation. These representations are stored as an array of geometrical shapes along with matrices that represent their orientation and relative positioning. The process of converting these representations into an image is called rendering. It is theorized that the human brain performs a process like inverse graphics, which is the opposite of rendering [5]. The brain might deconstruct a hierarchical representation of the visual information received from our eyes and match it with already learned patterns and relationships. Similarly, capsule networks aim to learn this representation.

The key idea of inverse graphics is that representation of objects does not depend on the view angle. In 3D graphics, a so-called pose may represent the relation between 3D objects, which includes translation plus rotation. In order to correctly classify and recognise objects, it is important to preserve hierarchical pose relationships between object parts. Capsule networks represent this data as a 4D pose matrix which incorporates relative relationships among objects.

Traditional CNNs incorporate pooling between layers to enable translation invariance and to prevent overfitting. To ensure the precise relationships of the 4D pose matrix are not lost, capsule networks use a novel dynamic routing algorithm between capsule layers, with no pooling. The first layer of a CapsNet learns to extract features by backpropagation and this output is fed to the capsules which learn weights similarly to an auto-encoder, see Figure 2.

2.3. Siamese Networks

Siamese neural networks are twin neural networks with identical weights that are trained together on pairs of input vectors. In practice, this arrangement may be accomplished by running two forward passes on a single network, then evaluating the two outputs. The loss function used in these networks is called the Triplet Loss or the contrastive loss. There are three parameters for triplet loss: the input image sometimes called the anchor image, a positive vector (true image) and

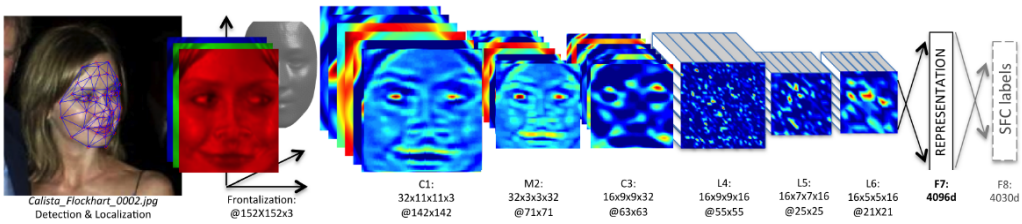

Figure 3. DeepFace Architecture by Taigman *et al*. [16]

a negative vector (false image). The positive vector acts as a regularizer to prevent overfitting and the negative vector forces the network towards learning the correct weights. Sometimes a margin is also added to the loss function to push the anchor-positive and anchor-negative pairs further apart from each other. Euclidean distance, Manhattan distance or cosine similarity are the standard distance measures used to compare encodings to each other. These networks are widely used in face recognition tasks most notably DeepFace [16] created by a research group at Facebook that recognises faces with an accuracy of 97.35% (Figure 3).

2.4. Self-Supervised Learning

Self-Supervised learning is a training technique that has been gaining traction over the past several years. In a typical self-supervised learning pipeline, a pretext task is defined which will generate labels from unlabeled data. These labels and the original dataset they were derived from are then used to pretrain a network.

The goal of a pretext task is to push the network to learn a structural understanding of the data. This way, the network is not as dependent on supervised training with the dataset's true labels. An example of this can be seen in Figure 4, from Gidaris *et al*. [2], where the image is flipped 0, 90, 180, and 270 degrees, and the goal of the network is to classify the amount by which the input image was rotated.

The overall goal of self-supervised learning is to eventually reach an accuracy from exclusively self-supervised training comparable to that which current state-of-the-art networks reach when trained entirely through supervised learning. This would allow researchers to leverage the massive amount of unlabeled data available on the internet. It was also remove the bottleneck in certain domains where labels are either extremely expensive or extremely difficult to obtain.

Recently, much progress has been made in the fields of Natural Language Processing and Computer Vision using self-supervised learning. Convolutional Neural Networks pretrained through self-supervised learning techniques are coming closer and closer to reaching the accuracy of networks trained in a purely supervised manner, but this is still an active area of research.

Another benefit of self-supervised learning is that, by designing the pretext task in a specific way, the network can be trained to learn a higher-level semantic understanding of a dataset. In certain cases, with the right pretext task, the filters learned by the net-

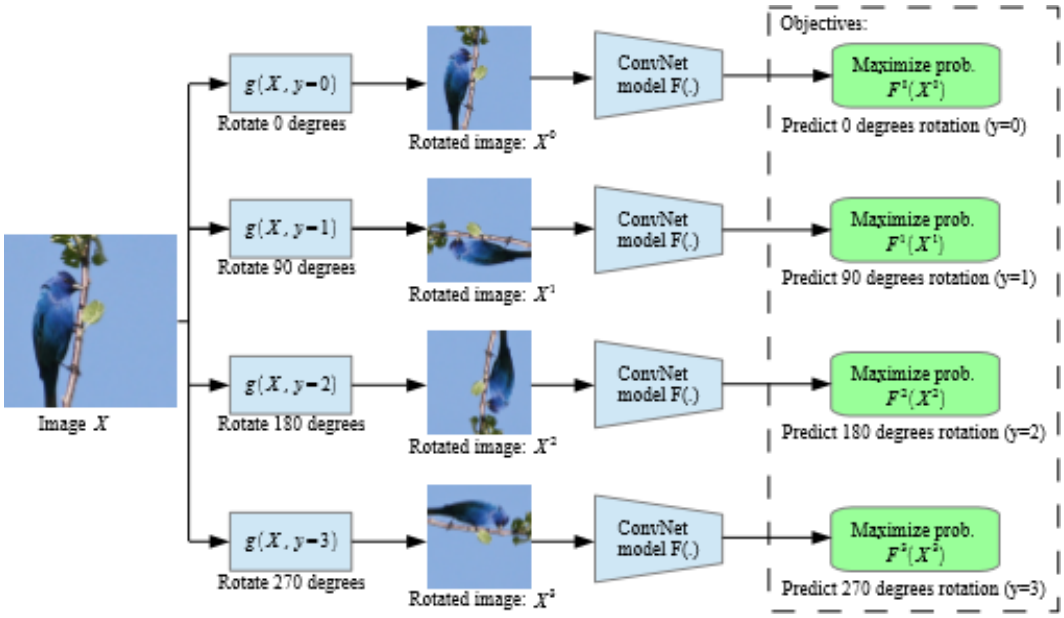

Figure 4. Example Architecture for Self-Supervised Pretext Task *et al*. [2]

work through pretraining are more versatile than those learned through supervised learning [2].

3. Methods

3.1. Datasets

Two different benchmark datasets for image classification were selected for this project. The first is the MNIST [9] dataset, which consists of 70,000 handwritten digits. The training set consists of 60,000 instances and the test set consists of 10,000 instances. Each image is a 28x28 matrix converted to grey-scale. The second dataset is The Database of Faces (AT&T). This dataset consists of 400 different faces. Each instance is a 92x112 matrix converted to grey-scale. Both datasets were downloaded from the GTDLBench [10] github repository, which contains several benchmark datasets used for image recognition and classification.

3.2. Multi-Layer Capsule Network

3.2.1 Network Architecture

The Multi-Layer Capsule network consists of an Encoder-Decoder network architecture which is identical to the original capsule network with the exception of augmenting the capsule layers. The first layer of the encoder network (Figure 5) is a convolutional layer which detects the basic features in the two dimensional image. The output feature map from this first layer is then passed onto the level-1 capsule layer. The job of the capsules in this layer is to take basic features detected by the convolutional layer and produce combinations of the features. The level-1 capsule layer is passed onto a level-2 capsule layer with 64 capsules. These capsules consist of a 64 dimensional vector and provide a higher level understanding of what features of the face are being learned. The

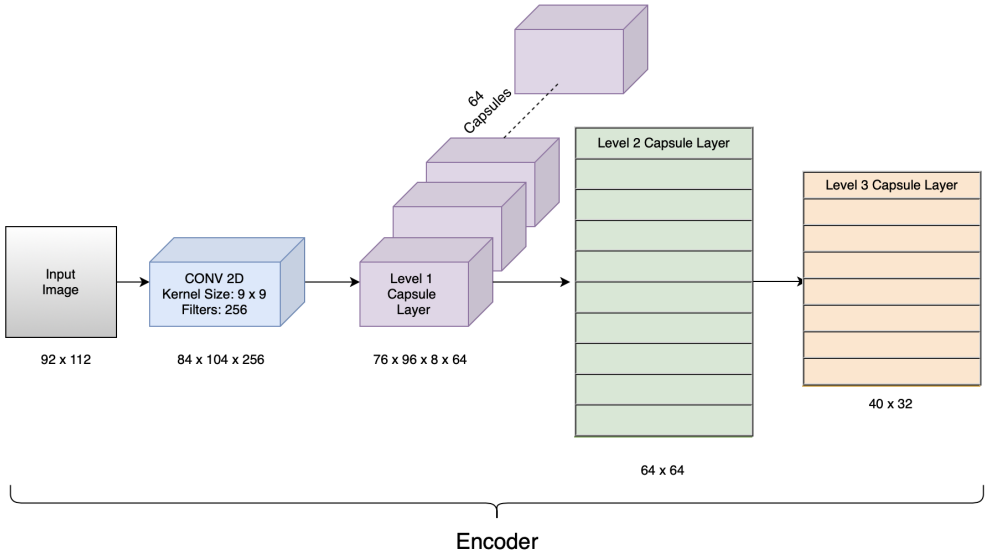

Figure 5. Architecture for the encoder of the Multi-Layer Capsule Network.

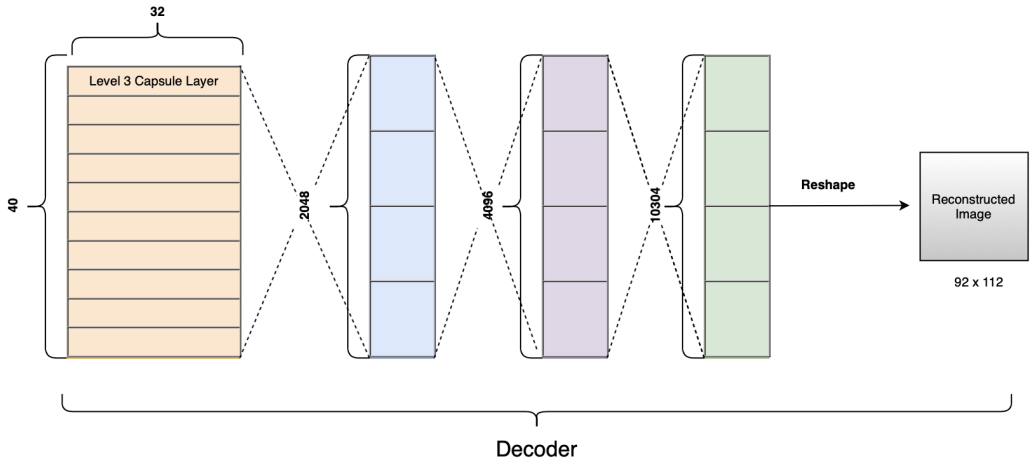

Figure 6. Architecture for the decoder of the Multi-Layer Capsule Network.

output of the level-2 capsule layer is passed onto the final level-3 capsule layer (or FaceCaps Layer) which consists of 40 capsules, each for one of the 40 distinct people in the AT&T dataset. Each of these capsules is a 32-dimensional vector where each dimension signifies some feature of the face for a specific person. While training, the output of the FaceCaps layer is passed onto a decoder network which is used as a regularizer and takes as input only the correct capsule vector while the remaining vectors are masked. The decoder network (Figure 6), hence, forces the capsule to learn the features that are important for recreating the original image. The decoder network consists of multiple (in this case 3) fully connected layers. The

last layer of the decoder is then reshaped to the size of the original image (92 x 112) and denotes the reconstructed image.

3.2.2 Marginal Loss Function

In the capsule network, the length of the instantiation vector signifies the probability of the capsule's entity being present at the scene. Down-weighting the loss for absent entities stops the learning from shrinking activity vector lengths for all entities. A marginal loss allows for multiple faces to be predicted and is used for each face capsule. The total loss is the sum of the losses of all the capsules. The loss function is given by,

$$T_k = \left\{ \begin{array}{l} 1, Label\ of\ class\ K\ present \\ 0,\ otherwise \end{array} \right\}$$

$$\begin{aligned} L_k &= \underbrace{T_k \max\left(0, m^+ - \|\mathbf{v}_k\|\right)^2}_{class\ present} \\ &+ \underbrace{\lambda\left(1 - T_k\right) \max\left(0, \|\mathbf{v}_k\| - m^-\right)^2}_{class\ not\ present} \end{aligned}$$

3.2.3 Reconstruction Loss Function

The reconstruction loss is simply a euclidean distance between the reconstructed image from the decoder and the corresponding input image. Thus, the closer the reconstructed image to the input image, the better. The reconstruction loss is multiplied by a regularisation factor so it does not dominate over the marginal loss.

$$Reconstruction\ Loss = ||Image - Reconstructed\ Image||$$

The total loss can be given as,

$$Total\ Loss = Margin\ Loss + \alpha * Reconstruction\ Loss$$

3.2.4 Latent Vectors Encodings

As seen in Figure 7, the decoder and the final layer of the encoder (Level 3 capsule layer) can be discarded to create an auto-encoder of a kind. Since the output of the intermediate capsule layer represents the positional relationships between detected facial features, flattening them into a single vector for the purpose of generating an encoding will not affect the instantiation parameters. At least one additional layer needs to be added to the original architecture since the initial capsule layer (Primary Capsule Layer) will focus on the understanding of basic image features whereas the secondary capsule layer (Digit Capsule Layer) provides a positional relationship between each of these basic features. However, the number of capsules in the secondary capsule layer is dependent on the number of classes in the dataset. Hence, an intermediate layer is required to provide better positional relationships by working on top of the initial capsule layer and be flexible enough so as to not affect the outputs predicted from the final capsule layer.

3.3. Capsule Vector Perturbation Analysis

Each capsule in the FaceCaps Layer is a 32 dimensional vector. By holding 31 dimensions constant and by causing slight variations in one of the dimensions, we can understand the subtle properties learned by that

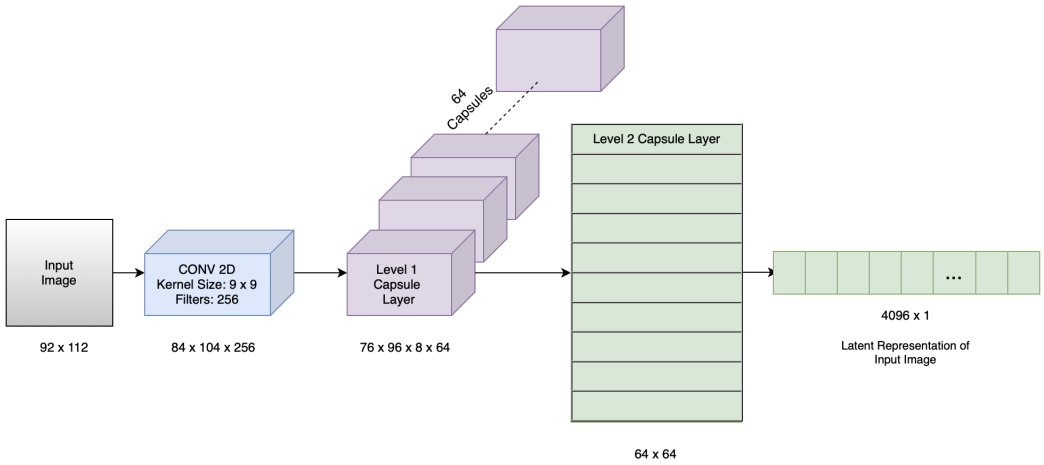

Figure 7. Latent vector generation using the encoder network.

dimension of the capsule. A similar analysis (Figure 8) is shown in Sara Sabour's paper for the MNIST dataset. There has been no formal analysis on facial image datasets, so we wish to reproduce similar understandings of the properties captured by the capsules with respect to facial features.

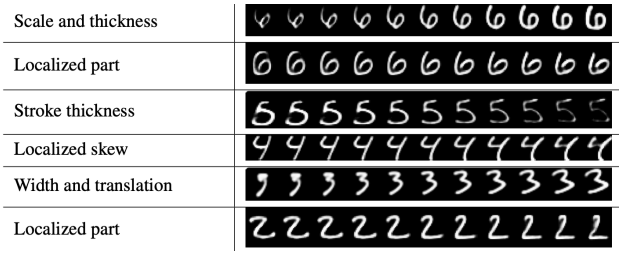

Figure 8. Dimension Perturbation as shown in Dynamic routing between Capsules by Sabour [11]

3.4. Multi-Layer Capsule Network Pretraining

In order to help the Multi-Layer Capsule Network gain a deeper semantic understanding of facial structure, the same architecture was pretrained using a self-supervised learning pretext task adopted from Gidaris *et al*. [2]. In this paper, the pretext task defined was to take every image in the training set and rotate them by 0, 90, 180, and 270 degrees. The amount of rotation was used to create four different class labels, and the network's task was to predict how much a given image had been rotated by. In order to successfully learn this task, the network had to gain a high-level understanding of the relationship between different objects in the picture. For instance, the network might learn to identify a dog through different features it evaluates, but if it can identify that the picture of the dog has been rotated upside-down, then the network has also learned what the proper orientation of the dogs body parts are, in space and relative to one another.

Because capsule networks are designed so that the low-level capsules capture information about different elements in the picture, such as their pose and orientation, this pretext task is an ideal candidate to push the Multi-Layer Capsule Network to learn a more detailed, semantic understanding of the human face, and be able to distinguish between different faces through

small inconsistencies.

Pre-training was performed using the same methodology as in Gidaris *et al*. The training set was augmented by rotating each image 0, 90, 180, and 270 degrees, and the architecture of the Multi-Layer Capsule Network was modified so that after the final capsule layer, three fully connected layers compressed the output of the capsule layer from a vector of length 160 to a vector of length 4. After this, the reconstruction was performed using a fully connected network. All loss functions and evaluation metrics were the same. This network was trained for 100 epochs, during which time it reached 100% training accuracy on the rotation pretext task.

3.5. Evaluation Strategy

Results of the original Capsule Network on the MNIST database were impressive, so the AT&T Faces database was chosen to evaluate the network on more challenging features. The goal was to better understand how capsules were representing identified features.

Each capsule in the FaceCaps Layer is a 32 dimensional vector. By holding 31 dimensions constant and by causing slight variations in one of the dimensions, we can understand the subtle properties learned by that dimension of the capsule. This process of Vector Perturbation Analysis is shown in Figure 8 from Sara Sabour's paper for the MNIST dataset. This paper attempts to reproduce similar understandings of the properties captured by the capsules with respect to facial features.

To evaluate performance of the Multi-layer Capsule Network versus the original network and CNNs, the latent vector encodings are used. This flattened 4096-dimensional vector provides a latent representation of the input face. Cosine similarity between vectors quantifies a score of how similar the network representation of each person was to other members of the data set versus other images of the same person.

Additionally, measures of classification accuracy were used to compare networks. For this measure an online poll was also conducted to collect human classification accuracy and confidence scores.

4. Results

4.1. Comparison with the Original Capsule Network for MNIST Dataset

To verify correct implementation of the network, the results of the Multi-Layer Capsule Network (MLCN) were compared to the original capsule network implemented on the MNIST dataset. The original capsule network was trained for 100 epochs and achieved a training accuracy of 100% (Figure 9). The model generalized well and achieved a test accuracy of 98.84%. The MLCN achieved a similar performance on the MNIST dataset with a Training Accuracy of 100% and a Testing Accuracy of 98.64%. The MLCN

Architecture	Accuracy	
	Training	Testing
MLCN	100.0	98.64
Original CapsNet	100.0	98.84

Table 1. Accuracy comparison on MNIST data.

starts off with a higher loss and achieves a lower training accuracy in the initial epochs as compared to the original network (Figure 10). However, the loss and accuracy are quickly optimized and show a similarity to the results of the original network.

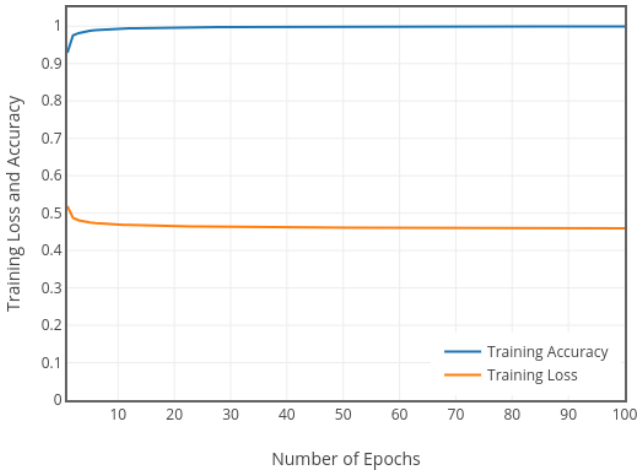

Figure 9. Training Graph of Original Capsule Network on MNIST Dataset.

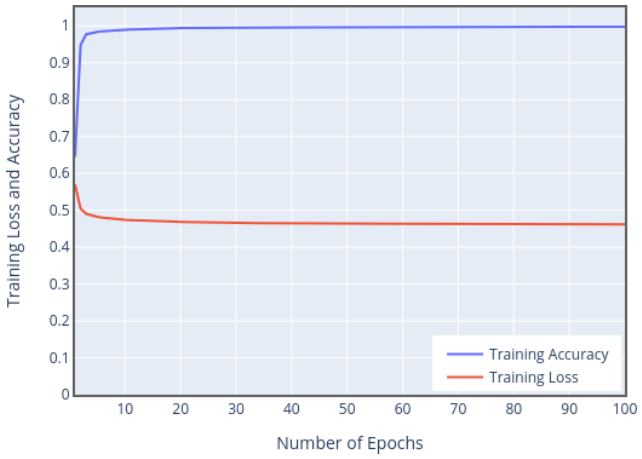

Figure 10. Training Graph of MLCN on MNIST Dataset.

The decoder of the MLCN generates a near identical reconstruction of the ground-truth image passed to it, as seen in Figure 11. This provides confidence that there is no downside to adding another layer to the capsule network.

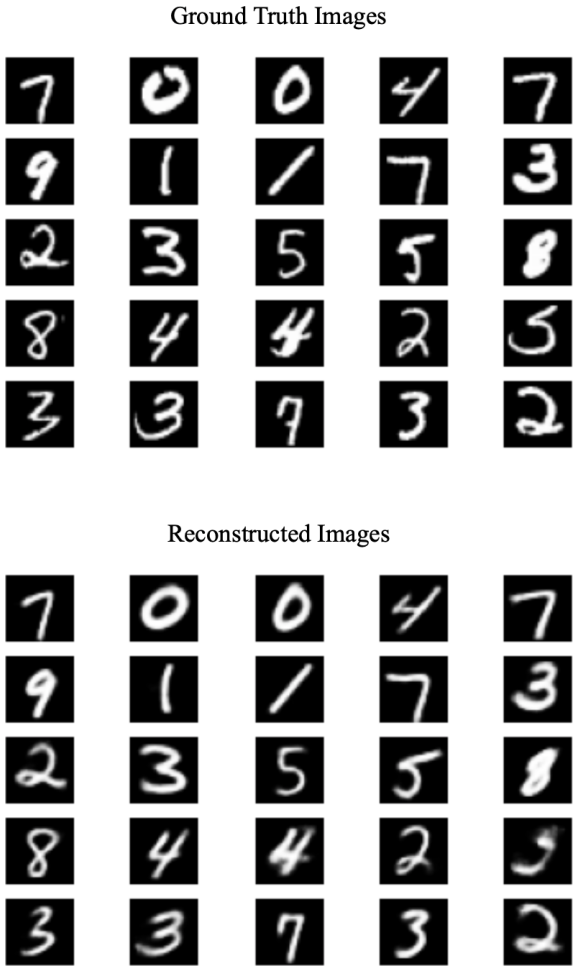

Figure 11. Ground Truth Images vs Reconstructed Images.

The latent vector generation shows that the additional capsule layer resulted in encodings that are good representations of the corresponding image. As seen in Figure 12, the encodings generated indicate an impeccable match by cosine similarity of the test image to its corresponding label. These results show the modified network works exceptionally well for the basic MNIST dataset.

Vector perturbation analysis was also performed to visualize what each of the capsule dimensions learn as shown in Figure 13. 15 dimensions were held con-

Figure 12. Latent vector encodings of MNIST, compared with Cosine similarity measure.

stant while varying the 16^{th} dimension by a factor of 0.05. This resulted in a visualization of each dimension's properties. Dimension 1 seems to be controlling the curvature of a number. Dimension 2 seems to be handling the thickness and stretch of a digit. What the dimension seems to be learning can be variable for each task, but the results are similar to those found by Sabour [11].

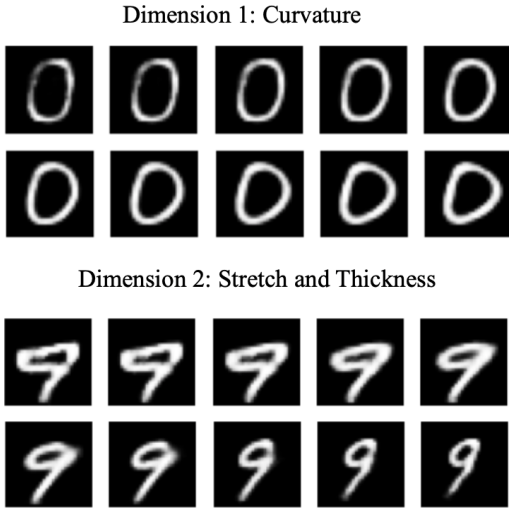

Figure 13. Vector Perturbation Analysis using Multi-Layer Capsule Network.

4.2. Multi-Layer Capsule Network - AT&T Faces Dataset

Having established the base case of the MNIST dataset, let's look at a much more complex dataset in the AT&T Database of Faces. The AT&T data set consists of 10 faces each belonging to 40 different people, hence a total of 400 images. We divided this data set into 320 images for training (8 images per person) and 80 images for the purpose of testing (2 images per person).

4.2.1 General Analysis

The Multi-Layer Capsule Network was trained for 800 epochs with a decline in its reconstruction loss. The initial reconstruction loss weighting was set to 0.005 so that the model could learn how to reconstruct the face better. Once the training reconstructions looked near identical at around 200 epochs the weight of the reconstruction loss was decreased to 0.0005 so as to increase the importance of classifying the faces correctly. This was further reduced to 0.0001 after 500 epochs. The effects of these variations can be seen clearly in the training graph (Figure 14) at the 200^{th} and 500^{th} epoch.

The network stabilized with a training accuracy of 100% and achieved a test accuracy of 80%. We believe the network could reach only 80% test accuracy since our training dataset is pretty small with a few samples per person. The network was compared to several other models, whose results are summarized in Table 2 and described in the remainder of this section.

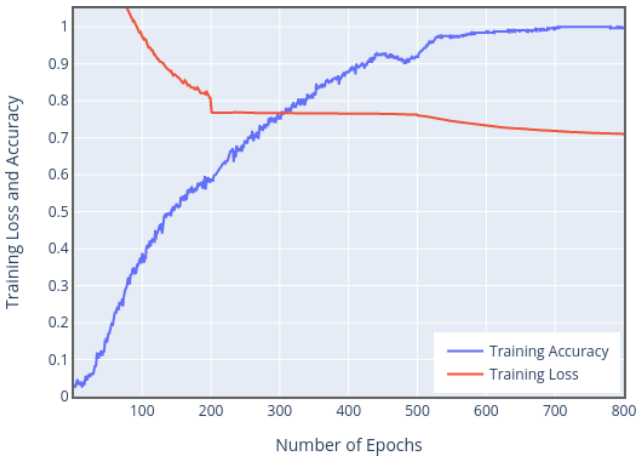

Figure 14. Training Graph for MLCN on AT&T Dataset.

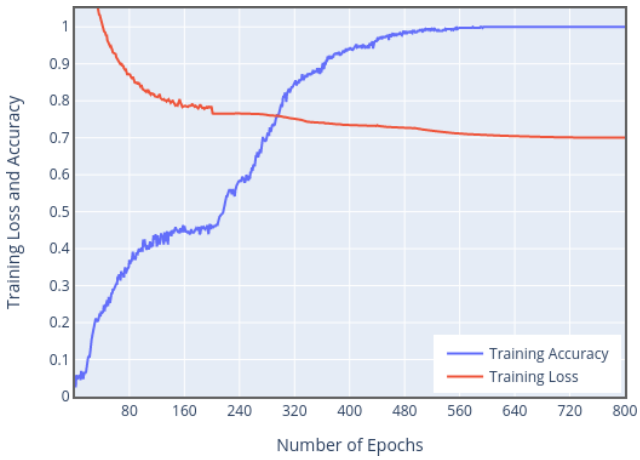

Figure 15. Training Graph for Pretrained MLCN on AT&T Dataset.

Architecture	Accuracy	
	Training	Testing
MLCN Baseline	100	80.0
MLCN Pretrained	100	88.8
Original CapsNet	100	78.8
Siamese CNN	100	81.2
FaceNet (vggface2)	-	100
FaceNet (casia-webface)	-	97.5
Human Online Survey	-	71.8

Table 2. Accuracy comparison on AT&T data.

4.2.2 Latent Vector Analysis

Our main goal through this network was to generate encodings by using the intermediate layer, see Figure 18 for example. The encoding generated by the MLCN architecture identified the test image about 80% of the time. The encodings seem to be able to give us a degree of identification, or a measure of similarity of the images. Cosine Similarity was calculated between latent vectors to quantify the degree of affinity between images.

4.3. Pre-Trained Multi-Layer Capsule Network - AT&T Faces Dataset

Pretraining the Multi-Layer Capsule Network with an image rotation task provided a boost in performance. After following the same training procedure as the baseline Multi-Layer Capsule Network, the pretrained version reached a test accuracy of 88.75% (Table 2).

In addition to this, the Pretrained Multi-Layer Capsule Network was able to correctly identify 5 out of the 10 people that the Baseline Multi-Layer Capsule Network could not.

Figures 16 and 17 show a visualization of the feature maps obtained when running one of the AT&T images through the convolutional layer of the Baseline Multi-Layer Capsule Network and the Pretrained Multi-Layer Capsule Network, respectively. Analyzing these feature maps can help give some interpretability to what the networks are learning. As can be seen from the figures, the networks appear to learn

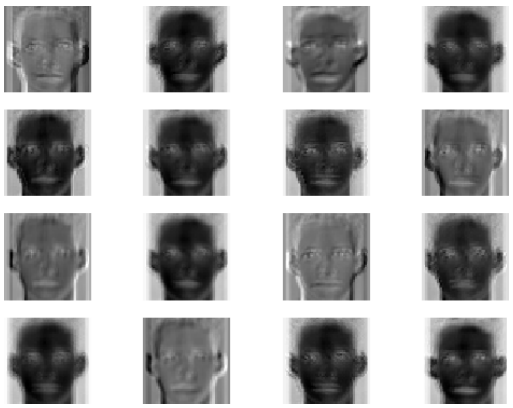

Figure 16. Visualization of feature maps from the convolutional layer of the Baseline Multi-Layer Capsule Network.

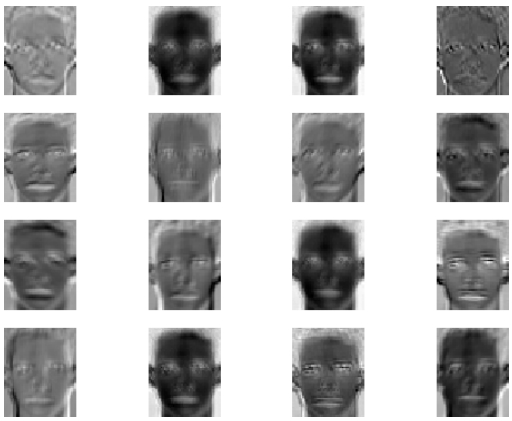

Figure 17. Visualization of feature maps from the convolutional layer of the Pretrained Multi-Layer Capsule Network.

different facial features at the lowest level by segmenting them with white lines (the areas on the feature maps that are most active). When comparing the two images, there is a greater density of white lines on the pretrained filters, which implies that the pretrained network is paying more attention to specific orientations and features on the person's face.

The feature maps obtained from the outputs of the primary capsule layer for both networks were too noisy to be interpretable from one another, but interpreting the outputs of the capsule layer's feature maps is an interesting direction for future research with capsule networks.

4.4. Comparison with other Architectures and with Human Performance

Performance was compared to two other networks designs. The first was a Siamese Covolutional Neural Network [3]. Each half was made up of three convolutional layers followed by three fully connected layers. It uses ReLu activation with batch normalization between layers. A contrastive loss function is used to train the network to differentiate pairs of images. It was trained with 100 batches of 64 images sampled from the training set, with approximately half of the pairs of images being from matching classes in each batch. Training accuracy was 100% and test accuracy 81%. This network served as a good comparison to the capsule network because it had a similar level of complexity. Comparing which images each network found to be most similar to each other helps to clarify how the capsule network is recognizing features.

The second network was an implementation [12] of the FaceNet (Inception-Resnet-v1) architecture [13]. This is a 22 layer deep network which trains with a triplet loss function. A model pre-trained on the large data set VGGFace2 (450,000 images of 10,500 individuals) yielded 100% accuracy on the AT&T data set with no additional training. A model pre-trained on CASIA-WebFace (3.3M images of 9000 individuals) had 97.5% accuracy.The nearly perfect results of the second network highlight the power of a large training set and a deep network which is the current standard

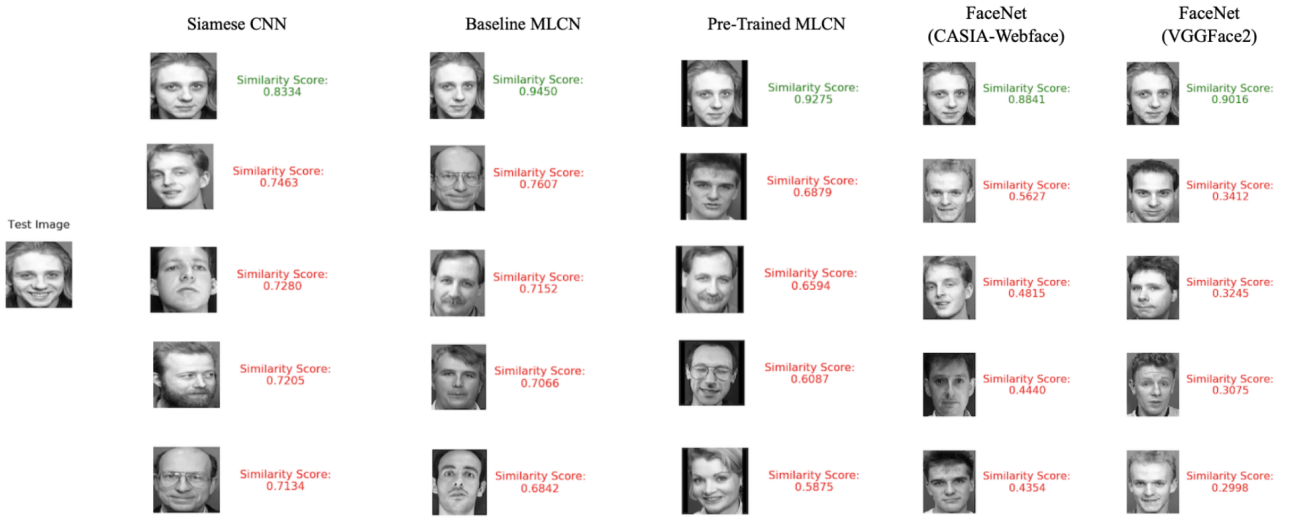

Figure 18. Similarity Measures for Label 3.

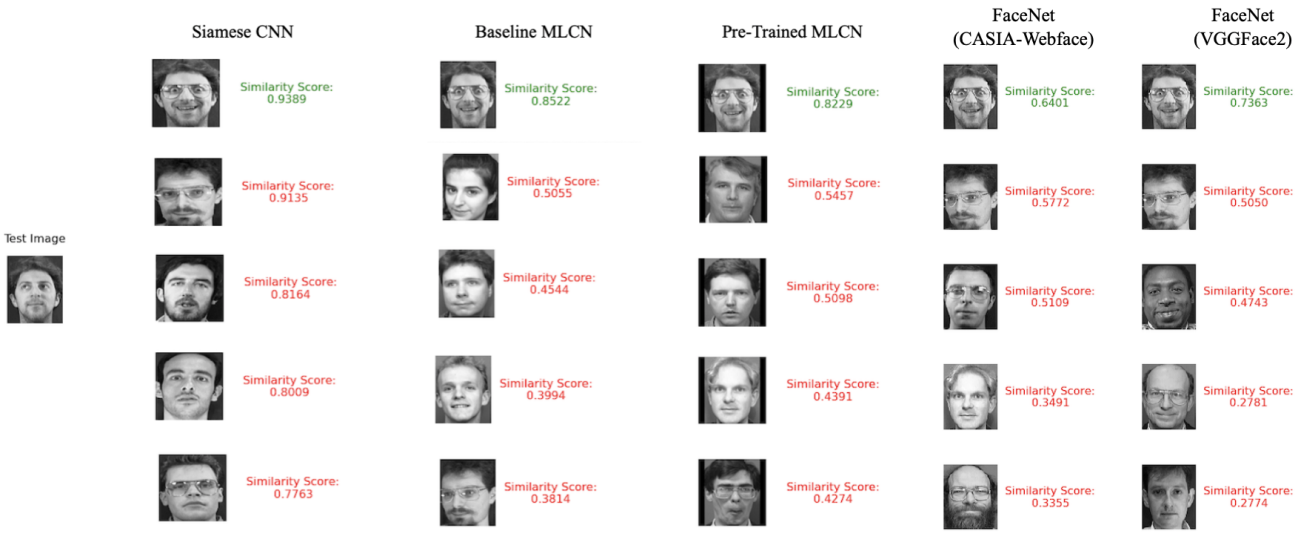

Figure 19. Similarity Measures for Label 37.

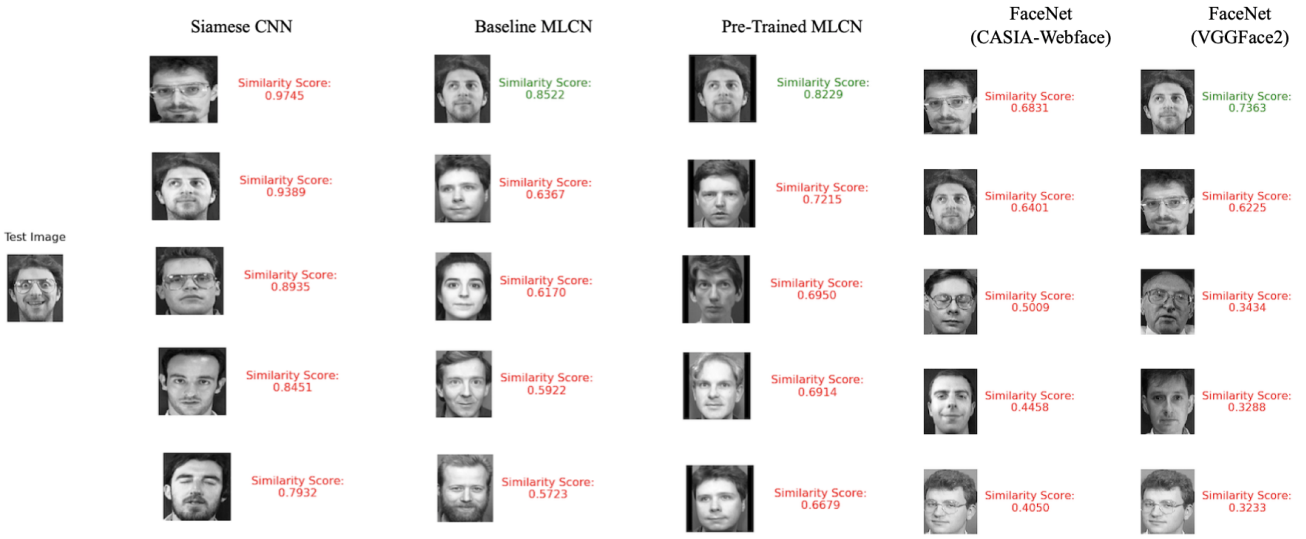

Figure 20. Similarity Measures for Label 37.

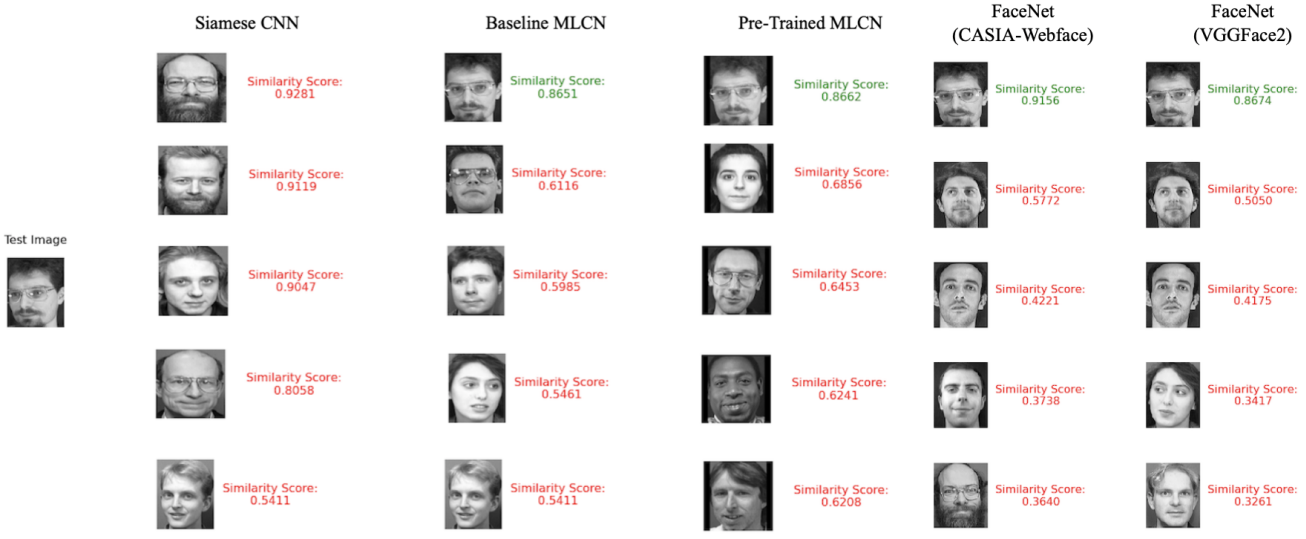

Figure 21. Similarity Measures for Label 8.

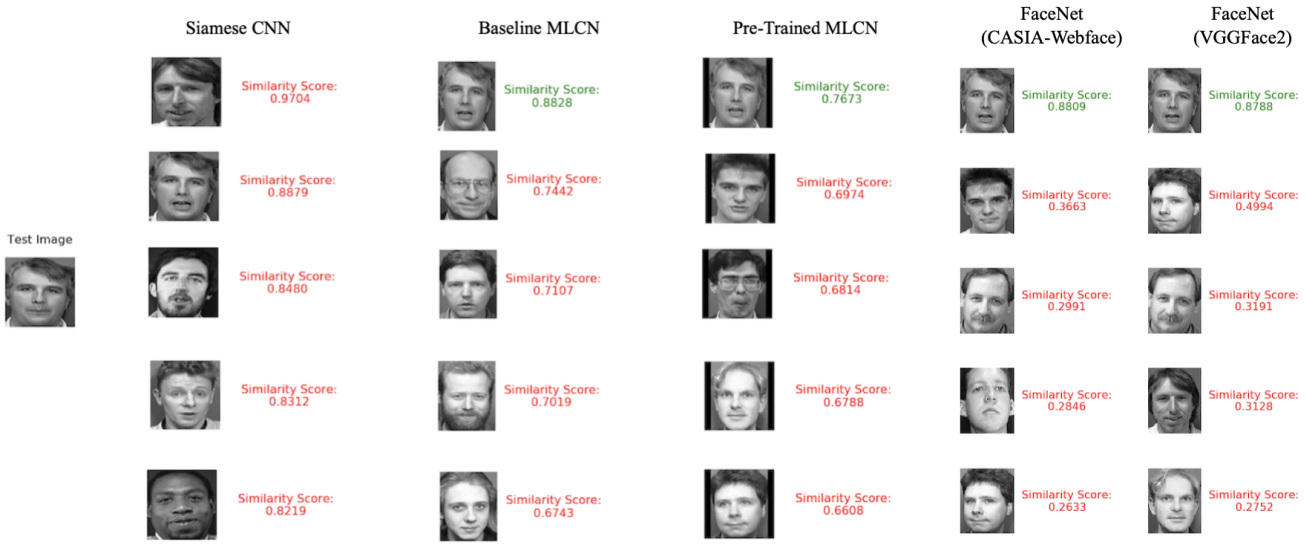

Figure 22. Similarity Measures for Label 6.

Architecture	Model Ranks				
	Label 3	Label 37	Label 37 (Inverted)	Label 8	Label 6
MLCN Baseline	1	2	1	4	1
MLCN Pretrained	2	3	2	3	4
Siamese CNN	5	1	Failed	Failed	Failed
FaceNet (vggface2)	3	4	3	2	3
FaceNet (casia-webface)	4	5	Failed	1	2

Table 3. Ranking of the models used in evaluation.

Architecture	Difference between the Cosine Similarities				
	Label 3	Label 37	Label 37 (Inverted)	Label 8	Label 6
MLCN Baseline	0.1803	0.3467	0.2155	0.2535	0.1386
MLCN Pretrained	0.2396	0.2772	0.1014	0.1806	0.0699
Siamese CNN	0.0881	0.0254	-0.0356	-0.5947	-0.0825
FaceNet (vggface2)	0.5604	0.2313	0.1138	0.3624	0.3794
FaceNet (casia-webface)	0.3214	0.0629	-0.043	0.3384	0.5146

Table 4. Difference between the Cosine Similarities of the top performing label and the true label.

in state-of-the-art networks.

Figure 19 and 20 show the top five images from each network with the highest cosine similarity score for class label #37. An online survey was conducted to get human responses for the same image. Interestingly, many of the same individuals mistakenly chosen by humans are top candidates for the networks as well (Figure 23).

Figure 23. Incorrect Responses from Human Survey

Tables 3 and 4 showcase the ranking of models and the difference in their similarities for each of the 5 tests (Figures 18 - 22). For the table of model ranks, The Baseline Model and the Pre-Trained model achieved near consistent top ranks followed by the FaceNet models and lastly the Siamese Convolutional Network model. For the table of cosine similarity differences, we take the difference of the top-performing similarity measure that is not the true label and the similarity measure of the true label. The higher this difference, the better the model. If this difference is negative, the true label was ranked lower than the top-performing label.

The Baseline MLCN and Pre-trained MLCN model had a consistent set of differences whereas the Siamese CNN and FaceNet(casia-webface) have a few instances of negative differences. The FaceNet(vggface2) model achieved higher values however this model has a lot of variation in its differences. The results from these two tables show our model's learning capability given its shallow depth and the number of images it was trained on. The limited size of the AT&T training set is addressed in the next section as one of the limitations of this paper.

4.5. Limitations

A major limitation throughout this study was the small size of the AT&T dataset. We believe this is the primary reason why we were not able to accurately reconstruct the images, as this is a task that is typically learned when trained on extremely large datasets such as ImageNet. This can be seen in Figure 25.

We believe the small dataset size was also the reason we were not able to get better results results for the vector perturbation analysis. This can be seen in Figure 24.

For the original perturbation analysis in Sara Sabour's paper, the vector perturbation was done using MNIST, which includes much simpler images than

Figure 24. Vector Pertubation Analysis for Dimension 31

the AT&T Faces dataset, and so the effects of perturbations were simpler to interpret. More data would allow the networks to generalize better, which would help us to see how perturbing different feature values affects their outputs.

Figure 25. Reconstructions for Baseline MLCN

Another limitation was the ambiguity of features learned by the capsule networks. By the second level, the feature maps had already been broken down into extremely small sizes, and saw it was difficult to visually interpret anything from the raw pixel outputs. Interpretation of results is a major challenge across all of deep learning, and due to the promise capsule networks have shown to identify important elements in images, researching more efficient ways to interpret capsule layers is an exciting potential direction of research.

5. Conclusions

For this study, we introduce Multi-Layer Capsule Network, an augmented version of the original Capsule Network architecture. We use this architecture to improve the representations learned by a Capsule Network's primary capsule layer by adding a second capsule layer in order to push the network to learn better representations in each capsule. We then further improve the representations learned by the Multi-Layer Capsule Network by pre-training it with a self-supervised pretext task. We evaluate our results by analyzing the cosine similarities found by the networks between pairs of photos from the AT&T Faces dataset, as well as by evaluating the networks' test accuracy values. The test accuracy values are compared against other state-of-the-art architectures such as the original CapsNet architecture and Siamese Convolutional Networks. Our Multi-Layer Capsule Network outperforms the original Capsule Network, and the pretrained Multi-Layer Capsule Network outperforms both the original CapsNet and a Siamese CNN. While neither of our networks reached the benchmark of 100% set by FaceNet, they still show impressive results for being trained on an extremely small dataset and having relatively shallow architectures.

6. Individual Tasks

Vikram Shenoy formulated the hypothesis, designed the network architecture and implemented the encoder half of the modified CapsNet. Alex Chowdhury implemented the decoder half of the CapsNet

and the pre-trained version. Both included coding and training the respective architectures, and making sure results were reasonable. Harry Hartenstine was responsible for selecting trained comparison networks that could be run on his local workstation, and comparing the results to the capsule networks. All group members documented their respective contributions in the paper and tabulated results and images.

References

[1] Kunihiko Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*, 36(4):193–202, 1980.

[2] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018.

[3] Harshvardhan Gupta. Facial similarity with siamese networks in pytorch. https://github.com/harveyslash/Facial-Similarity-with-Siamese-Networks-in-Pytorch, 2017.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[5] Krizhevsky A. Jaitly N. Tieleman T. Tang Y. Hinton, G. Does the brain do inverse graphics? *Brain and Cognitive Sciences Fall Colloquium*, 2, 2012.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Yann LeCun, Corinna Cortes, and Christopher Burges. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/, 1998.

[10] Ling Liu, Yanzhao Wu, Wenqi Wei, Wenqi Cao, Semih Sahin, and Qi Zhang. Benchmarking deep learning frameworks: Design considerations, metrics and beyond. *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pages 1258–1269, 2018.

[11] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules, 2017.

[12] Davod Sandberg. Face recognition using tensorflow. https://github.com/davidsandberg/facenet, 2018.

[13] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015.

[14] J. Su, D. V. Vargas, and K. Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[16] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *IEEE Conference on Computer*

Vision and Pattern Recognition, pages 1701–1708, Columbus, OH, 2014.