STAT 479 Project: Face to Painting

Lingfeng ZHU, Zhuoyan XU, Cecily LIU

UW-Madison, Department of Statistics

April 2019

Introduction

Here is what we mainly did in this project:
First: Compare CNNs with basic Machine Learning models to show the advantages of DL
Second: Given a human face photo, our model will:

- 1.Generate painting stylistic portraits from this photo.
- 2. Perform face recognition to identify this person.

Motivation

- Many applications of face recognition
- Photo recognition and Face verification
- People's addiction to photo effects and filters
- Applications: Improvement in face recognition; Painting filters for photo apps

Dataset

We use this dataset to compare the results of some ML methods and CNNs:

Use icrawler to scrape the images by ourselves. Totally 6000 images of 15 celebrities, such as Gal Gadot, Robert Downey jr etc.

Gal Gadot Robert Downey jr Scarlet

Figure 1: Google image search

Data Augmentation

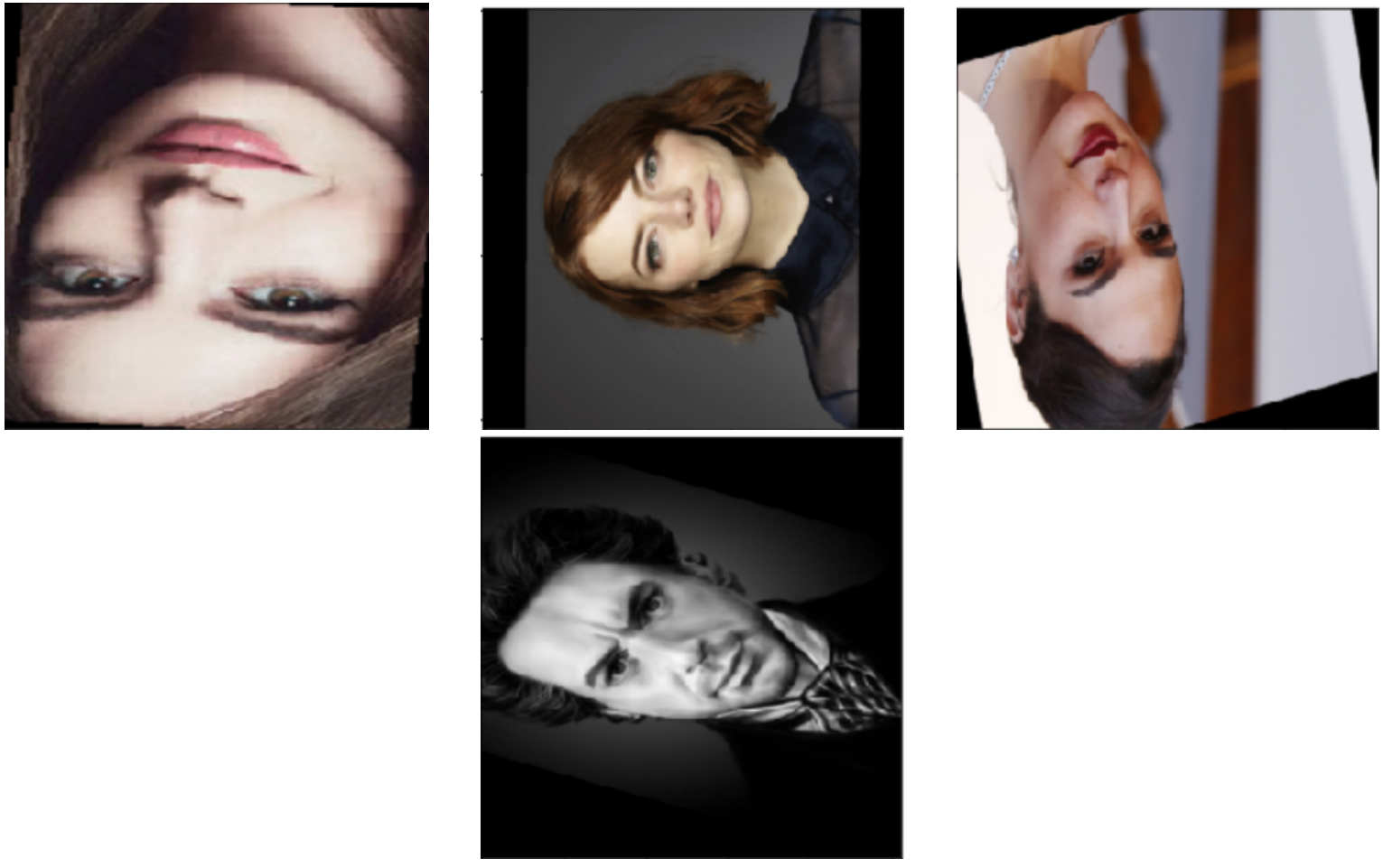


Figure 2: Data Augmentation

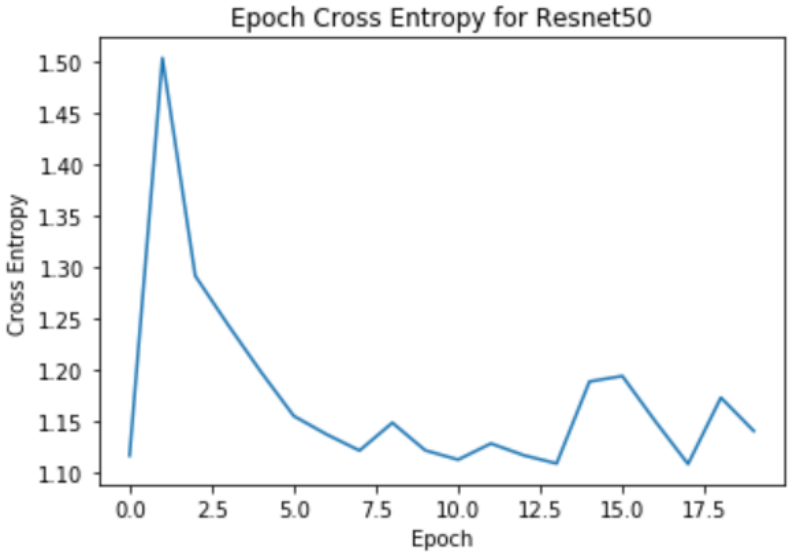
Transfer learning

- VGG16: Freeze all the Conv layers + 3 FC layers, train 2 FC layers + an additional output layer.
- VGG19: Freeze all the Conv layers + 5 FC layers, train 1 FC layers + an additional output layer.
- ResNet50: Freeze all Conv layers, train the output layer.
- ResNet101: Freeze all Conv layers, train the output layer.

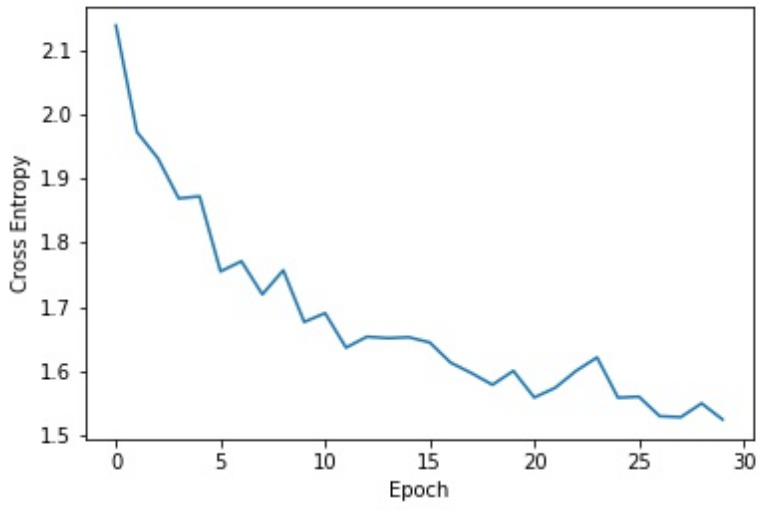
Comparison

Model	Test accuracy
SVM	25.02%
GBDT	20.13%
VGG16	38.12%
VGG19	34.26%
ResNet101	35.88%
ResNet50	40.00%

Cost through epoch



ResNet50



ResNet101

Figure 3: Epoch loss

Neural Style Transfer

- Based on the pre-trained model, we can perform neural style transfer to generate portrait paintings from a human face image.
- The basic idea of style transfer was from [Gatys et al., 2015. *A neural algorithm of artistic style*].
- Some methods of this project were from Coursera online courses by Andrew Ng.

VGG 19

We choose the pre-trained VGG 19 model to perform a transfer learning.

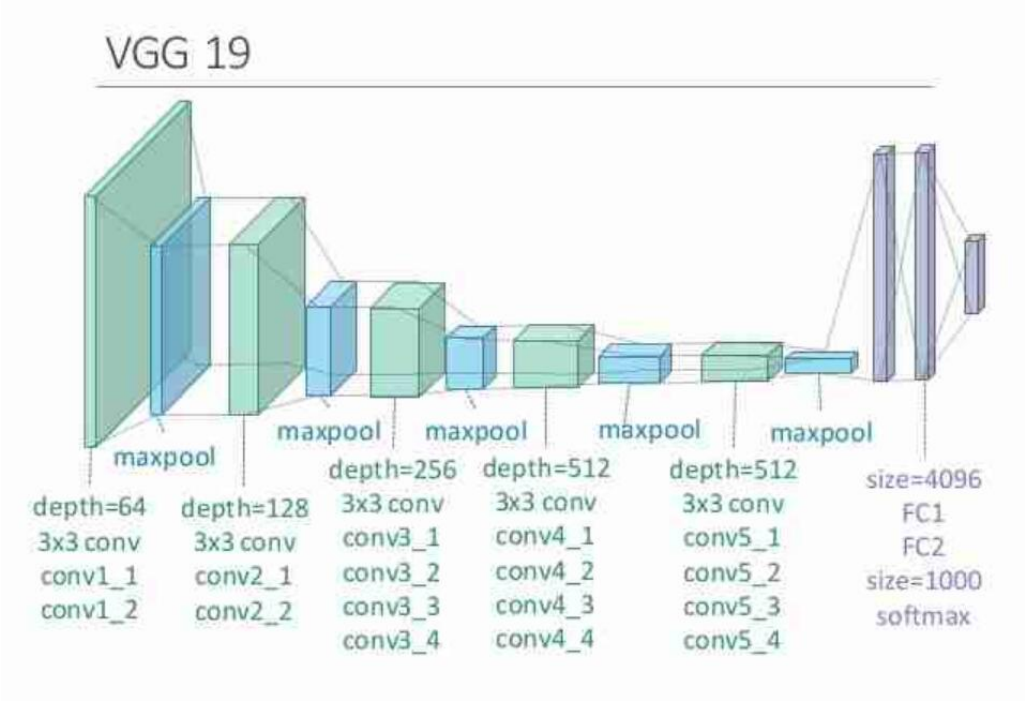


Figure 4: Construction of VGG 19

[Simonyan & Zisserman 2015. *Very deep convolutional networks for large-scale image recognition*]

Neural Style Transfer

Neural Style Transfer is one of the applications of deep learning. As seen below, such method can merge two images to a new generated image.

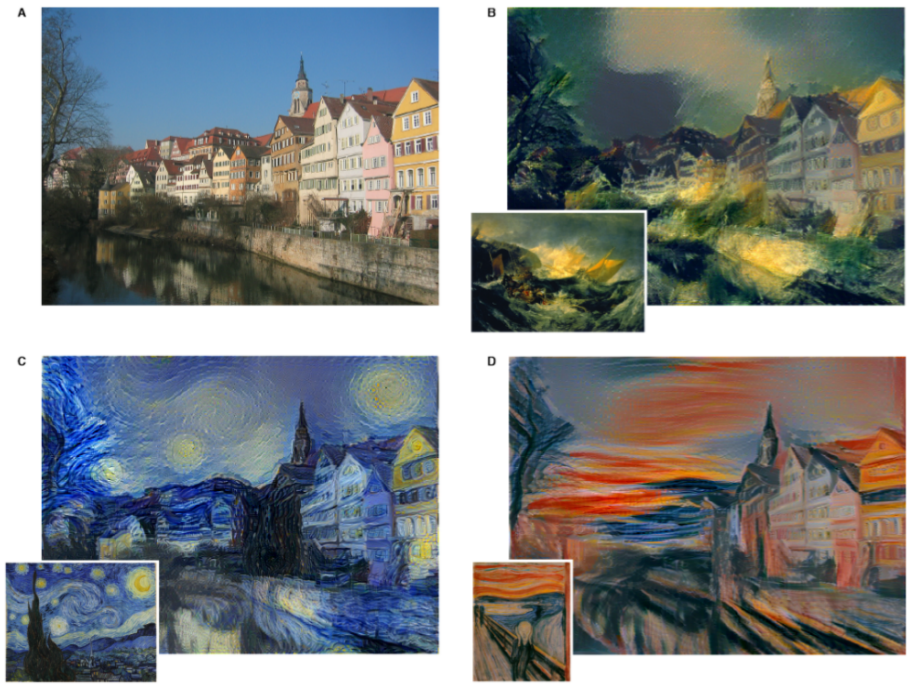


Figure 5: Examples from the paper

[Gatys et al., 2015. *A neural algorithm of artistic style*]

Content Image

First, we choose one image of Gal Gadot as the content image. We will use the pre-trained CNN model to generate one portrait painting from this content image.

Figure 6: Content image: Gal Gadot

Content Image Cost

Let C denote this content image, and G denote our generated image. We can choose some layers in the middle of our network, say, layer l, our goal is to make the generated image G have similar content as the content image C. To do so, we can define the content cost function as:

$$Cost_{\text{content}}(C, G) = \frac{1}{4 \times n_H \times n_W \times n_C} \sum_{\text{all entries}} \left(a^C - a^G\right)^2$$

Where n_H, n_W and n_C denotes the height, width and number of channels of layer l. a^C and a^G denote the activations of hidden layer l if we set C and G as input of our pre-trained network, so, they are $n_H \times n_W \times n_C$ tensors.

Style Image

Then we choose some style images to set the style of our painting:
For example, the starry night by Van Gogh as our style image S.

Figure 7: Style image: Starry Night

Style matrix: Gram matrix

To compute the style image cost, we need to compute the style matrix. Consider layer l as an example, if we unroll the activation of layer l into a matrix V with $(n_H \times n_W)$ columns and n_C rows, the style matrix is defined as:

$$W^{[l]} = VV^T$$

W is a $n_C \times n_C$ matrix. The value W_{ij} measures the similarity of the *ith* filter and the *jth* filter of this layer l.

Style Image Cost

Similar as content cost, if we set S and G as the input of our network respectively, we can calculate the corresponding style matrices W^S and W^G. Then, the style image cost of layer l is defined as:

$$Cost_{\text{style}}^{[l]}(S, G) = \frac{1}{4 \times n_C^2 \times n_H^2 \times n_W^2} \sum_{i=1}^{n_C} \sum_{j=1}^{n_C} \left(W_{ij}^S - W_{ij}^G \right)^2$$

Style Weights

To make our output better, we can calculate the style cost for many layers and compute the weighted average style cost:

$$Cost_{\text{style}}(S, G) = \sum_{l} \lambda^{[l]} Cost^{[l]}_{style}(S, G)$$

Where $\lambda^{[l]}$ is the weight of layer l. In this project, we choose five layers (conv1-1, conv2-1, conv3-1, conv4-1 conv5-1) from the network and set all the weights to be 0.2.

Total Cost

Now, we can then define the total cost to optimize:

$$Cost(G) = \alpha Cost_{\text{content}}(C, G) + \beta Cost_{\text{style}}(S, G)$$

We set weight parameters α and β as:

$$\alpha = 10$$

$$\beta = 40$$

We choose the layer conv4-2 from the network to compute the content cost.

Initialize Generated Image

We can initialize the generated image by adding some noise to the content image.

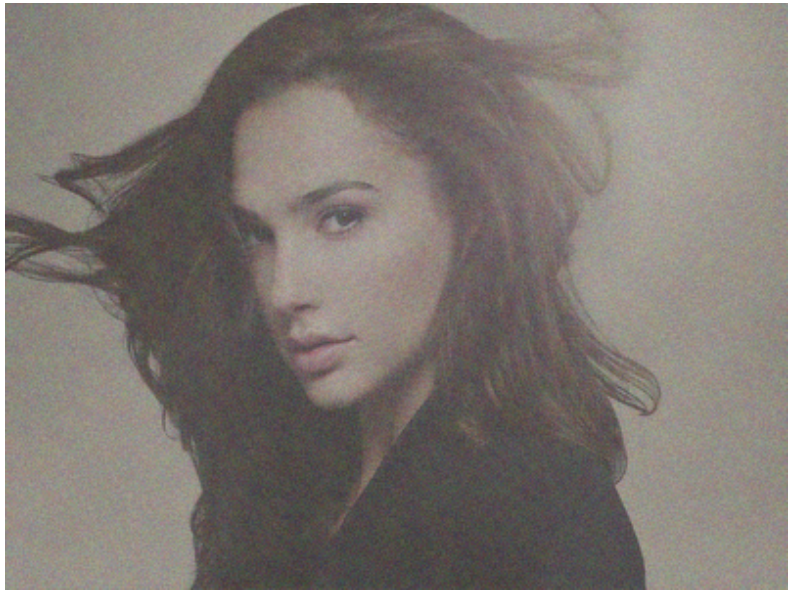


Figure 8: Initialized image

Train the model

Use the total cost function above, set the generated image as the input, we can train the VGG 19 model: but this time, we train the generated image instead of the parameters in the network.
This this the output of our model after 200 epochs:

Figure 9: Generated image

More Results

Here are some more results based on different style images:

Monet painting

Model Result

Figure 10: Monet Style

More Results

Sandstone

Model Result

Figure 11: Sandstone Style

More Results

Skrik

Model Result

Figure 12: Skrik Style

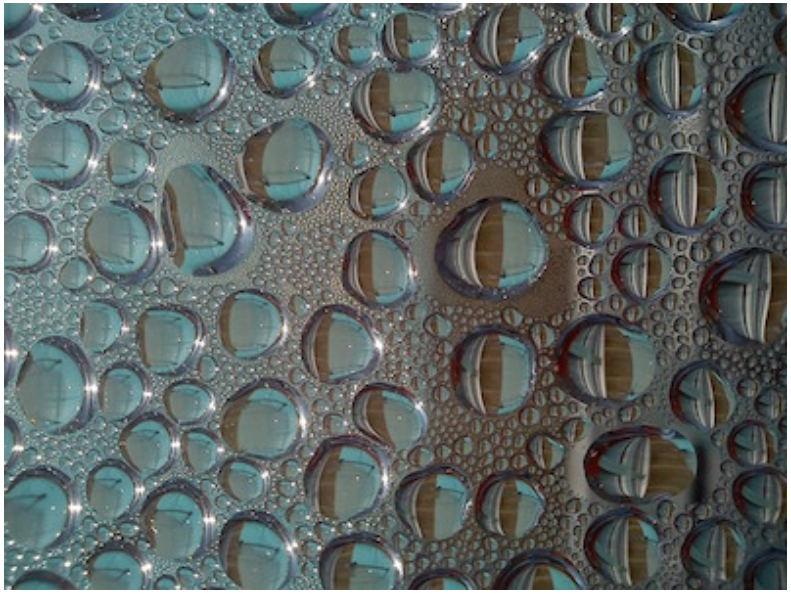
More Results

Stone

Model Result

Figure 13: Stone Style

More Results



Water Drop

Model Result

Figure 14: Water drop Style

More Results

Guernica

Model Result

Figure 15: Guernica Style

FaceNet: A unified embedding

- Learns a mapping from image to a Euclidean space as feature vectors where can measure similarity.
- When get feature vectors, we can use some classification method or just measure the distance between two certain images.
- The basic idea is based on [*Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." 2015*].

Dataset: LFW

LFW: Labeled Faces in the Wild, a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web.

Figure 16: LFW data

[http://vis-www.cs.umass.edu/lfw/]

Model structure

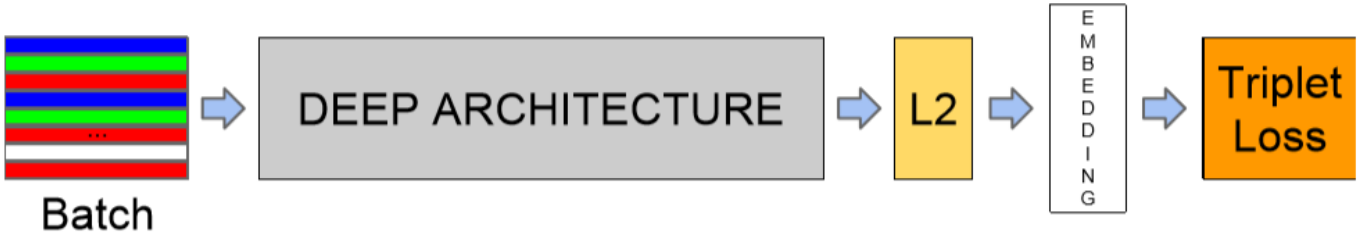


Figure 17: general model[Schroff,F etc.*Facenet: A unified embedding for face recognition and clustering*,2015]

Use Batch input + deep CNN + L_2normalization to get face embedding(feature vector), then we use triplet loss to train model. As we got the normalized feature vector, we multiply it by 10 follow the idea by [Rajeev, R etc.*L2-constrained Softmax Loss for Discriminative Face Verification*,2017].

Triplet loss

Figure 18: Triplet loss[Schroff,F etc.*Facenet: A unifiedembedding for face recognition and clustering*,2015]

We want:

$$\left\|x_i^a - x_i^p\right\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \forall \left(x_i^a, x_i^p, x_i^n\right) \in \mathcal{T}$$

where α margin, $\mathcal{T}$ is the set of all possible triplets in training set.

Triplet loss

Thus, the general loss function is

$$L = \sum_{i}^{N} \left[\left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2 - \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2 + \alpha \right]_+$$

Note: Most triplet will Not contribute to model and result in slow convergence.

Triplet selection

Hard (Margin) triplet to accelerate convergence. Given anchor x_i^a, select:

- hard positive: $\text{argmax}_{x_i^p} \left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2$
- hard negative: $\text{argmin}_{x_i^n} \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2$

drawback:

- infeasible to compute through whole dataset.
- mislabelled and poorly imaged faces would dominate the hard positives and negatives.

Triplet selection

To avoid it, we use "Generate triplets online" as suggested in the paper: select the hard positive/negative exemplars from within a mini-batch.
In practice:

- train with all anchor-positive pair in a mini-batch, and still select the hard negatives. All anchor-positive method was more stable and converged faster at the beginning of the training.

Triplet selection

According to the paper, selecting hardest negatives can lead to local minimum. Use *semi-hard exemplars* to mitigate it:

$$\left\| f\left(x_i^a\right) - f\left(x_i^p\right) \right\|_2^2 < \left\| f\left(x_i^a\right) - f\left(x_i^n\right) \right\|_2^2$$

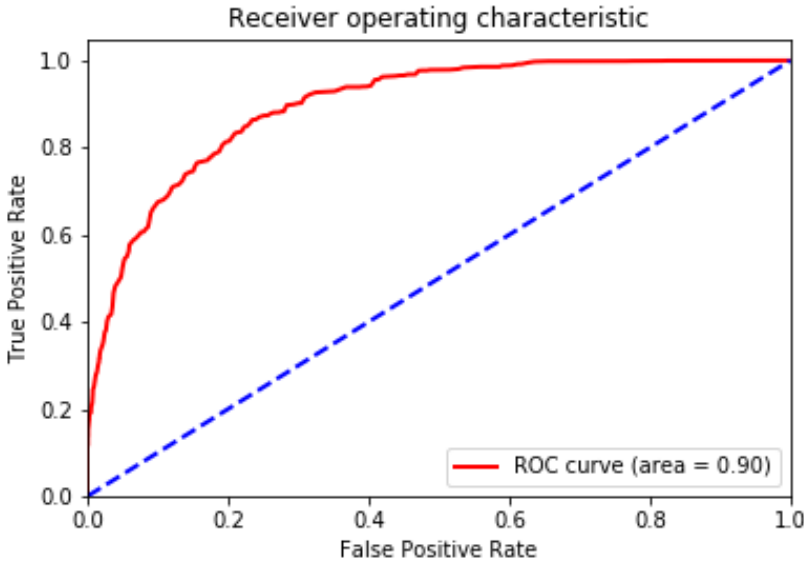
The semi-hard negatives is further to the anchor than positives, but still hard since anchor-negative distance is close to anchor-positive distance.
Note:**Correct triplet selection is crucial to fast convergence.**

Model Results

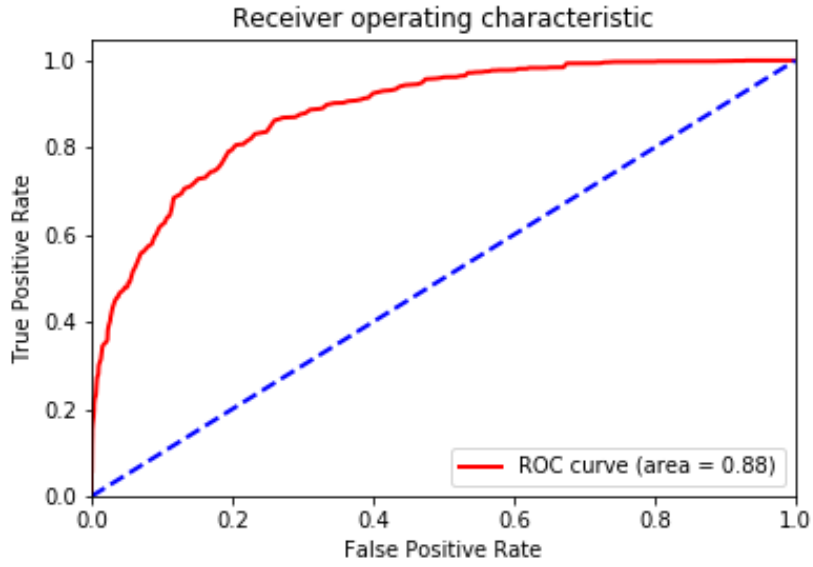
The CNN model results are as following:

CNN	Train accuracy	Test accuracy	AUC
Resnet34	86.10%	81.47%	0.90
Resnet50	82.40%	79.87%	0.88
VGG 16	88.77%	80.22%	0.88
VGG 19	88.00%	80.03%	0.88

Resnet ROCs



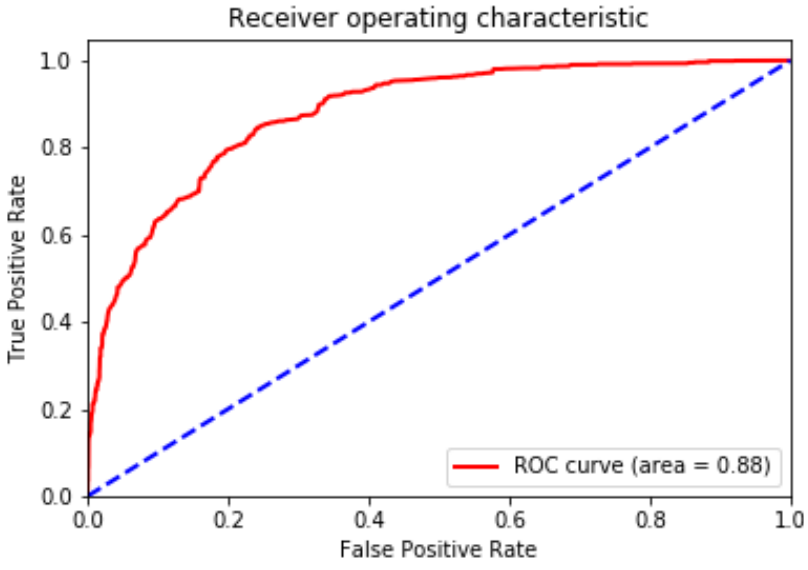
Resnet34



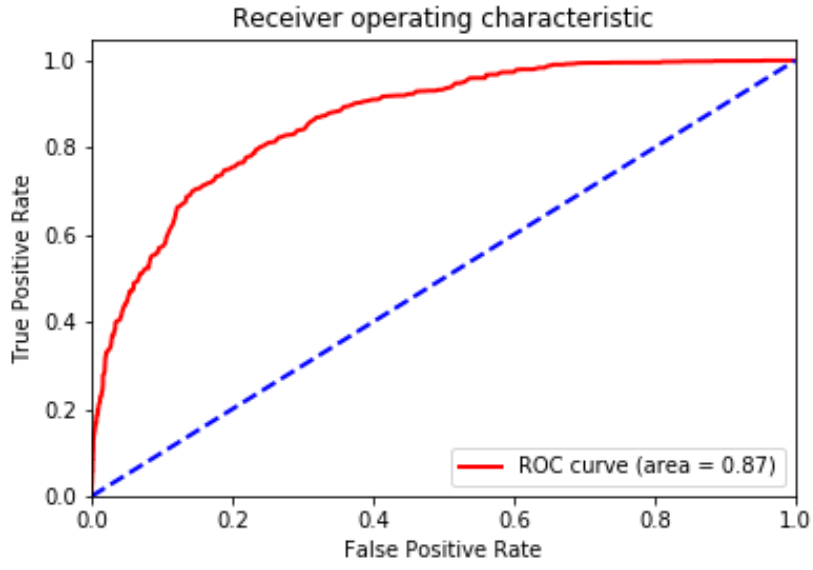
Resnet50

Figure 19: Resnet ROCs

VGG ROCs



VGG16



VGG19

Figure 20: VGG ROCs

Triplet loss

4 triplet loss plots (of the VGG19) in this slide. (2 for minibatch test train; 2 for epoch test train)

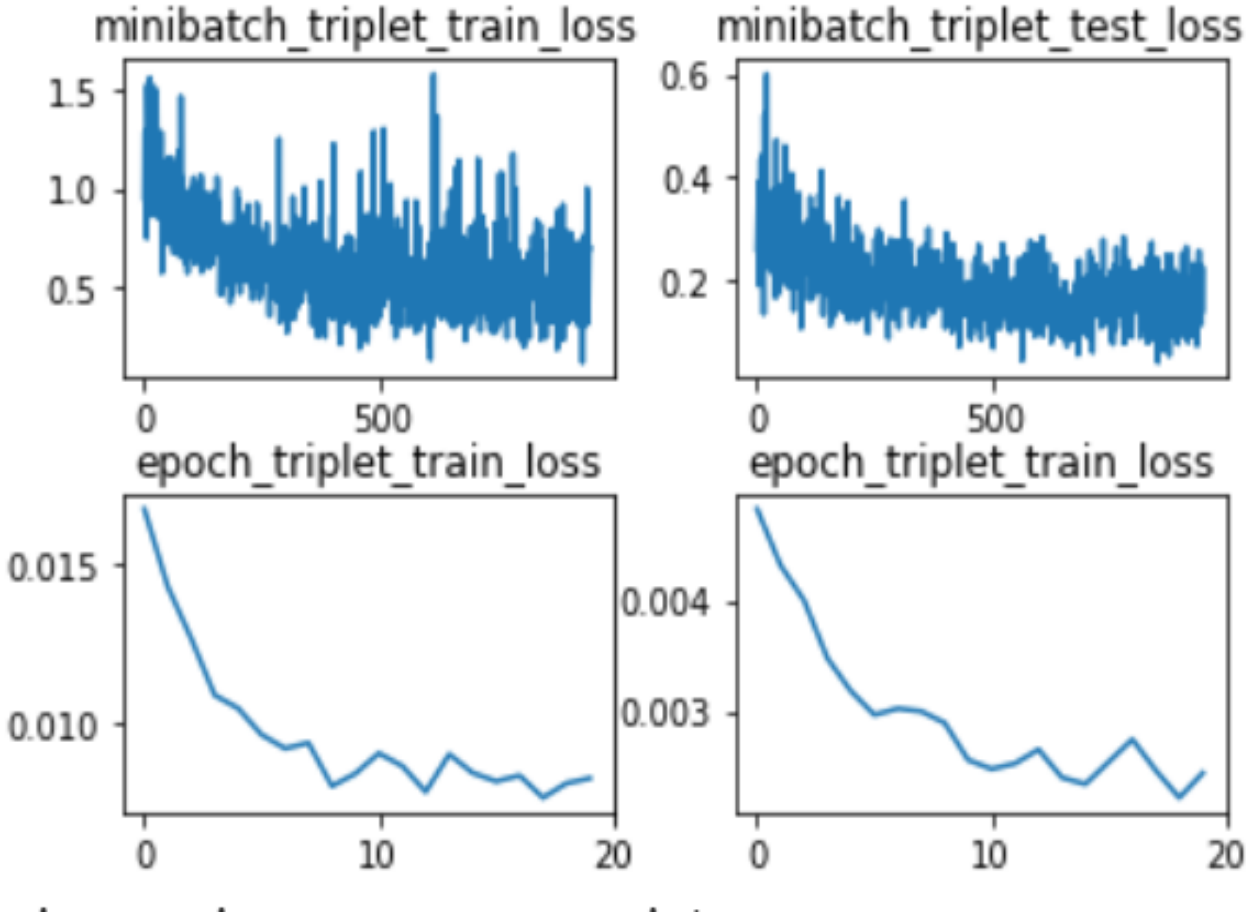


Figure 21: VGG19

Bibliography

1. Simonyan & Zisserman 2015. *Very deep convolutional networks for large-scale image recognition*
2. Gatys et al., 2015. *A neural algorithm of artistic style*
3. Schroff,F etc.*Facenet: A unified embedding for face recognition and clustering*,2015
4. Rajeev, R etc.*L2-constrained Softmax Loss for Discriminative Face Verification*,2017.
5. He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep residual learning for image recognition,2016*
6. https://github.com/tbmoon/facenet
7. https://www.deeplearning.ai/

Thank you

Thank you for your attention!