ConvNets for Super-Wide Baseline Matching

Colin Wei, under mentorship of Amir Zamir
colinwei@stanford.edu

Abstract

In this project, I explore various convnet architectures for two verification problems: face verification, and baseline matching. Face verification is a problem for which convnets have been applied extensively and have produced extremely good results; I attempt to take similar network architectures and apply these results to the baseline matching problem, training on a dataset of streetview images.

1. Introduction

Baseline matching is essentially matching corresponding points in pictures of the same object taken at different viewpoints. Super-wide baseline matching is performing baseline matching for larger differences in angle between viewpoints. Although humans can easily perform super-wide-baseline matching, computers cannot do this very well - popular techniques such as SIFT are not very accurate as the differences in between viewpoints increases by too much. This project takes a data-driven approach to baseline matching, by attempting to train a convolutional neural network on the image set. This is an approach that has not been taken before, as there has not been enough data in the past to allow such an approach.

I am working on this project with Amir Zamir and other members of the CVGL lab; my role in the project is taking care of the deep learning aspect while they will be in charge of other aspects such as data collection, etc. Since applying convnets has not been attempted for this problem before, I split up the deep learning in two parts: the first was to make sure that I can use convnets to replicate results from similar problems such as face verification, and the second to begin work on the baseline matching problem itself.

2. Problem Statement

For the main part of the project, the basic problem will be to classify two input images as depicting the same object or viewpoint, or different object. The dataset we plan to train the neural network on will be taken from Google StreetView images, and it will be comprised of many pictures of buildings from different angles and different camera closeness. The problem will be to see whether we can map points from one image of the same building to a different image taken from the same viewpoint. The hope is that the convolutional neural network will be able to determine whether such a mapping exists, and accurately describe the mapping if there is one.

The problem I have worked on most for this project is replicating face verification results. Face verification is the problem of, given two images of people, determining whether those images are of the same person or not. I experimented with different network architectures described in various face verification papers.

3. Background/Related Work

While convnets have not been applied to baseline matching before, there has been a lot of past work on face verification producing state-of-the-art results. One popular approach to the face verification problem is first training a network on face recognition - classifying the identity of a person over a fixed set of identities in the training set. These networks are then used as facial feature extractors that are fed into a verification network that determines whether the faces are the same. In [6], Taigman et. al train a classifier to identify faces from a dataset of 4.4 million faces and 4030 identities. They apply alignment by detecting key points in the face - center of eyes, nose, and mouth - using which the image is cropped and translated onto fixed locations for each of the key points. They also apply 3D alignment to the face in order to account for out-of-plane rotations. Alignment helps their results heavily - without it, their test accuracy on the benchmark LFW dataset drops to 87.9 percent. Using the feature extractors gained from classification, they train a Siamese network that achieves 97 percent accuracy on the LFW dataset.

In [3], Sun, et. al propose a different architecture that also achieves very good results on the LFW dataset. Whereas most verification papers seem to use some sort of variant of a Siamese network - taking two distinct networks, extracting a feature vector for both images in the pair, and then comparing the two feature vectors to classify, Sun, et. al apply a different base architecture. To feed an image pair into a convnet, they simply stack the two images on top of each other, and then train directly off of that. They train 12 groups of 5 convnets which this method, which then feeds into a RBM that aggregates the outputs of all these convnets and performs prediction. Each of these groups of convnets takes droppings of different regions of the face images. With their methods, they achieve around 92 percent accuracy on the LFW dataset, using a much smaller training set than the DeepFace team. Sun, et. al also use alignment, though they only align around 3 points.

While the general architectures in the papers above seemed the most transferrable to the baseline matching problem, there were other architectures I considered. In [1], Chopra et. al propose a Siamese network that trains using a contrastive energy-based loss function, which minimized the energy of face pairs belonging to the same people and maximized the energy of dissimilar face pairs. In [4], Sun et. al take a similar grouped-convnet architecture, but this time train their network through classification too by classifying over 10000 identities. Using this method, they achieve an accuracy of 97 percent on LFW, on a much smaller dataset than the one used by [6].

One major difference between my attempts at face verification and the state of the art results listed in the papers above is that in general, I used less complicated network architectures (only training one stacked image convnet instead of 60), and fed the unaligned face images directly into my network. I did not attempt to align the faces before passing them into the convnet because alignment is a feature of face verification only, as every face has several key regions. However, when working with baseline matching, alignment is not an option since the alignment between the two images is exactly what we are trying to learn.

4. Technical Approach

4.1. Machines/Framework

I trained my networks on Amazon EC2 GPU's using the Torch7 framework. My code for training the networks and loading data was based heavily off of the fbcunn Torch package; I found Torch7 in general to be very fast.

4.2. Network Architecture

I experimented with two core architectures: a Siamese architecture, and a stacked image architecture. For the baseline matching problem, I plan to experiment between several network architectures. One option, the option I plan to explore first, is a Siamese architecture. Face verification papers such as [1], [4], and [5] all use some variant of the Siamese architecture. The Siamese network consists of two identical convents, followed by a layer that connects the output from the two different convnets and a loss function layer. Two images will be passed into the Siamese network as the input, each down one of the two identical convnets. For the stacked image network, I used 5 convolutional layers in the following layout, where the input is simply two images stacked on top of each other along the RGB depth channel:

input	128 by 6 by 128 by 128
conv5	128 by 40 by 128 by 128
reLu	128 by 40 by 128 by 128
pool	128 by 40 by 64 by 64
conv5	128 by 60 by 32 by 32
reLu	128 by 60 by 32 by 32
pool	128 by 60 by 16 by 16
conv3	128 by 100 by 16 by 16
reLu	128 by 100 by 16 by 16
pool	128 by 100 by 8 by 8
conv3	128 by 160 by 8 by 8
reLu	128 by 160 by 8 by 8
pool	128 by 160 by 4 by 4
conv3	128 by 200 by 4 by 4
reLu	128 by 200 by 4 by 4
pool	128 by 200 by 2 by 2
fc	128 by 2
binary softmax	prediction

Each stacked image network I trained was some variant of the above network, though I made small variations in between networks such as adding dropout layers, changing the input size to 6 by 64 by 64, etc. For the siamese networks, i used a similar architecture as one of the identical towers on either side of the network without the last fully connected layer, and combined the inputs from either tower in a subtraction layer:

output left	output right
output left - output right	
fc 128 by 800 to 128 by 2	
softmax and prediction	

5. Experiments

5.1. Face Verification

5.1.1 Data

For my training dataset, I used the largest labeled faces dataset I could find available - that was the Cross-Age Celebrity Dataset, which contains 163446 images of 2000 different celebrities across different ages. The dataset consists of close up shots of the faces of the celebrities, such as The type of image found in the CACD dataset is extremely similar to those in the LFW dataset, because they both consist of photos of celebrities in the public. For the CACD dataset, I created my training set by taking all images of people who in CACD that were not in LFW. In total, this resulted in around 120000 remaining images. I then con-

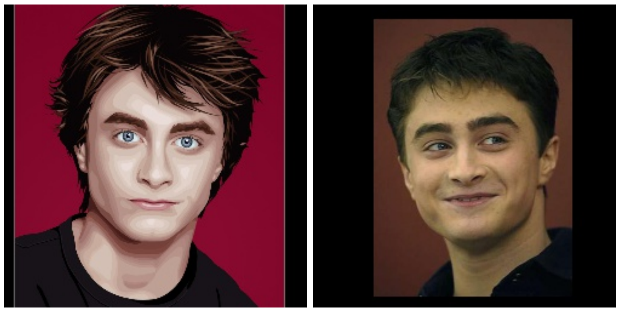
Figure 1. Typical images in the CACD dataset

structured training examples as follows: for each celebrity in the dataset, I generated a permutation vector whose length matched the number of that celebrity's images. I then paired consecutive images in the permutation vector and every other image in the permutation vector together for a training sample of same-id images, and for each same-id pair, I randomly generated a different celebrity and chose an image from that different celebrity in order for a negative training example. This way, the training set would have the same number of matches as incorrect matches in ID. This feature is extremely important so that when the vector trains, it does not get biased by the image set when predicting labels. For my test/validation set, I used the LFW dataset, which contains 13000 images of celebrities. I only used the subset of images in the LFW test set, for validation, which has 1000 images. The images in both datasets were 250 by 250; before passing them into the network, I downsized them into 128 by 128 images so that the network could run faster.

5.1.2 Training

During training, my primary objective was to decrease the loss function, as I noticed that this was very difficult to do for the binary problem and thus a good indicator of how well the network was learning. For face verification, I trained two network types: a image stack network, and a Siamese network, both on binary softmax classifiers. The batch size that I used for training was 128. Because I found that hyperparameter search was not very conclusive for binary softmax - it would take too long for different choices of hyperparameters to register different results - I tuned my parameters by hand, decreasing the learning rate when I found that the loss function was starting to plateau.

5.1.3 Results

Training the Siamese network did not work at all - I found that even after training the network for around 50000 iterations on a batch size of 128, there was no decrease in the loss function whatsoever - it hovered around 0.69, the loss obtained by random guessing. Even when I tried overfitting the network to a dataset of size 128, it took several thousand iterations on a batch size of 128 to show any improvement.

Training the image stack network worked significantly better than the Siamese network. Training for the image stack network went slowly too - on average, for a batch size of 128, it took around 3000 iterations training a model from scratch in order to notice any decrease in the loss of the network. The chart above demonstrates the change in loss over

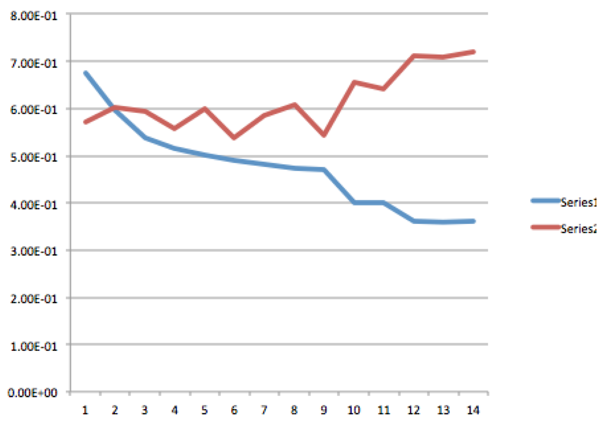

Figure 2. Loss of training set in blue, test set in red per epoch

the training set in blue, and the change in loss over the test set in red over each epoch. From the above graphic, it is clear that the convnet overfit very severely on the training set. The cause for this problem could also be the cause for why the Siamese network performed so badly from scratch: the gradients produced by binary softmax are so small that adding regularization terms in the first few epochs completely outshadow the direction of the gradient; perhaps the lack of regularization in the first few stages of training is why such severe overfitting occurred. This is because gradients from binary softmax are on the order of 10000 times smaller than the same gradients would be from classifying a problem such as ImageNet, since there are roughly 10000 more classes in ImageNet and thus random guessing in ImageNet would lead to a much stronger gradient signal. In the end, my image stack classifier achieved a best validation accuracy of 71 percent on the LFW dataset, which is significantly smaller than validation accuracies from state of the art results. However, those results also used significantly larger datasets or heavy alignment and data augmentation in their models, which I did not implement.

5.2. Baseline Matching

5.2.1 Data

For the baseline matching problem, I used a dataset of around 90000 images obtained from StreetView cameras. These images are of buildings, and are classified by location of the target point of the image. The goal on this dataset is to learn which images have the same target points, and which ones have different ones. For example, two same location image pairs in this dataset would look like where

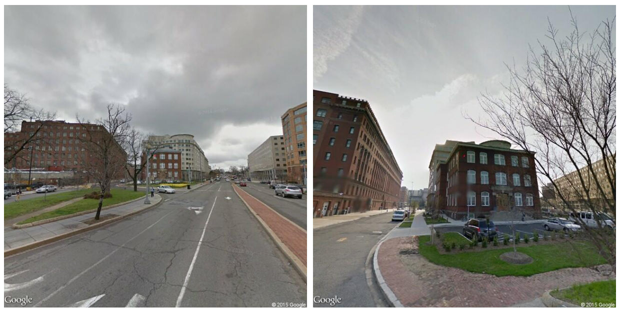

Figure 3. Example images

the center point of each image is approximately the target of the image. For my training set, I used a subset of around 60000 of these images, and my test set consisted of the other 30000. For both training and testing, I considered only the center 128 by 128 section of each image - since the camera is trained on those center patches, those centers would have the most similarity for same-target images and differences for different-target images. After cropping these images, I downsampled to 64 by 64 in order to speed up training on the dataset. For baseline matching, I split up the dataset into pairs the same way I split up the CACD dataset, generating 4 total pairs per image - 2 with same target location, and 2 with different target location.

5.2.2 Models

I tried out three different models on this dataset: the image stack model, a Siamese network fine-tuned on a classification network, and a Siamese network fine-tuned on a regression network. Though the image stack model again worked the best and the two Siamese networks did not learn very well, I still believe the fine-tuned classification model could be extremely effective. For my linear regression model, I implemented the following layers on the bottom of a Siamese network:

output left	output right
output left - output right	
linear regression	

where output left and output right are both 5 dimensional vectors, and I regressed them with a ground truth 5d vector. For an image of target location $t \in \mathbb{R}^3$, and cameras at location $c_1, c_2 \in \mathbb{R}^3$, with a angle of degree θ between the vectors $c_1 - t$ and $c_2 - t$, I let the 5-dimensional vector $f(c_1, c_2, t) = \left(c_1 - c_2, \frac{|c_1 - t|_2}{|c_2 - t|_2}, \theta\right) \in \mathbb{R}^5$. I chose to use this vector for regression because we have $c_1 - c_2 = (c_1 - t) - (c_2 - t)$, so the vector $f(c_1, c_2, t)$ essentially encompasses all the information about the transformation between $c_1 - t$ and $c_2 - t$. To generate the data to train this model, I took all possible pairs of images that have the same target location, and calculated the vector $f(c_1, c_2, t)$ for those images, as it does not make sense to calculate the transformation between two images not of the same target location. After training the model on regression until its learning curve plateaued, I trained a Siamese network by removing the last linear regression layer of the model and training on a fully connected layer from 5 nodes to 2.

To train a classification based model on this dataset, I again used knowledge about the target locations of the models. There were approximately 11000 target locations for around 60000 images in the training set; I simply trained a classifier on these images for these target locations, the same method used in a lot of papers about face verification such as [6]. The network architecture looked like:

standard convnet	128 by 100 by 1 by 1 last layer
fully connected	128 by 100 to 128 by 11135

To train a Siamese network on top of this, I used the architecture

classifier left without fc	classifier right without fc
classifier left - classifier right	
fully connected 128 by 100 to 128 by 2	
softmax loss	

5.2.3 Training

Although my Siamese networks made progress on the first layer of training, they did not work very well at the binary classification stage. However, they did learn extremely well in the classification stage - with a loss function that has been steadily decreasing:

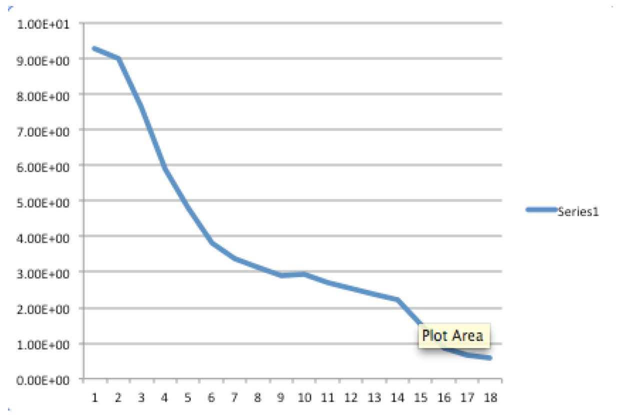

Figure 4. Loss function per epoch

I decreased the learning rate at the 14th iteration, hence the increase in the speed of convergence. While much of the speed of this convnet's learning is due to the fact that the training set is very small relative to the capacity of the net, it still stands this type of network trains significantly faster than binary softmax networks do. However, the improvements this network made did not appear during fine-tuning - the loss of the softmax was still stuck at 0.69, the loss achieved by random guessing. I believe this may be again due to overfitting - every time I started fine-tuning the network, the loss would blow up initially to around 10. This may be due to the fact that the network has already learned some biases, causing it to classify very wrongly.

For the regression classifier, my model plateaued very quickly to a loss not much lower than the original loss - it seems much less potent a model than the classification one. Furthermore, I believe that given a choice between a linear-regression based model and a classification based one to fine tune on, the classification based model would always work better because it relates better to the current problem of verification at hand.

Results from the image stack network obtained through baseline matching were very similar to my results on the same network from face verification. I also trained using the exact same network architecture for both models; the only difference was that the baseline matching network used input sizes of 64 by 64 in row/height, while the face verification network used input sizes of 128 by 128 in row/height. I got validation accuracy per epoch of

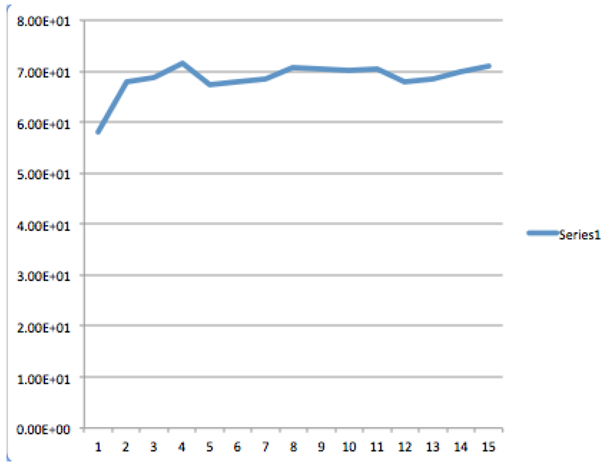

Figure 5. Validation accuracy per epoch

Again, this model suffered from substantial overfitting, as the two loss curves initially align and then branch off sharply. Furthermore, accuracy on the validation set plateaus really quickly because of overfitting. However, the size of each epoch is 3000 instead of 10000 for the face verification network - this means this network trained significantly faster than the face verification network. I believe that this is due to more alignment in these inputs, as I am cropping out a center square of 128 by 128 to train on. Since target locations are given to be the same and are located in the center, this cropping is very useful for helping the model.

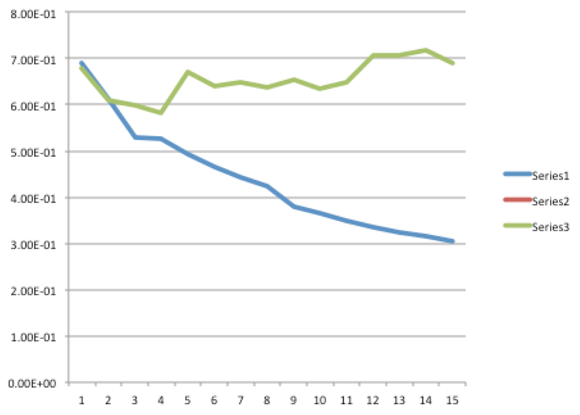

Figure 6. Loss per epoch

6. Conclusion

Since I am continuing work on this project through next quarter, I feel that my current work has been a good step in scaling everything up for training on larger datasets. The first major conclusion I can draw from this for future work is that binary softmax is not very robust, as training on larger and larger datasets and models seems to increase the variance in the data really quickly and thus slow down the effectiveness of the model's learning. Therefore, if I plan to train from scratch using binary softmax, I should have some method of adding robustness to my classifier - in [3], Sun et. al did this by aligning the face images, grouping crops of the images together, and training on those. Indeed, it does seem that the image stacking classifier responds very well to more alignment - It seems that training binary softmax on un-augmented data can still give results, but not very robust ones.

I believe that fine-tuning on classification is a very powerful approach that can give good results; I plan to go back through my implementation of fine-tuning on classification across target location and figure out how to make it work. Either there is a bug in my code, or Siamese networks are very brittle - I don't know which is the case, but I hope to figure that out. Given how hard it is to train a binary softmax classifier versus how much quicker a classification network learns, I believe that it will be very useful to be able to fine tune a binary softmax classifier instead of training from scratch.

7. Acknowledgements

Thanks to Amir for providing me with the data and giving me advice for the project!

8. References

[1]. Chopra, Sumit, Raia Hadsell, and Yann LeCun. ”Learning a similarity metric discriminatively, with application to face verification.” Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[2]. Krizhevsky, Alex. ”One weird trick for parallelizing convolutional neural networks.” arXiv preprint arXiv:1404.5997 (2014).

[3]. Sun, Yi, Xiaogang Wang, and Xiaoou Tang. ”Hybrid deep learning for face verification.” Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE, 2013.

[4]. Sun, Yi, Xiaogang Wang, and Xiaoou Tang. ”Deep learning face representation from predicting 10,000 classes.” Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.

[5]. Khalil-Hani, Mohamed, and Liew Shan Sung. ”A convolutional neural network approach for face verification.” High Performance Computing and Simulation (HPCS), 2014 International Conference on. IEEE, 2014.

[6]. Taigman, Yaniv, et al. ”Deepface: Closing the gap to human-level performance in face verification.” Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.