DenseBag: a Neural Network Architecture for Gaze Estimation

Marcel Bühler
Data Science Masters, ETH Zürich
buehlmar@ethz.ch

Brian Regan
Data Science Masters, ETH Zürich
bregan@ethz.ch

ABSTRACT

This report presents DenseBag, a neural network architecture for the problem of eye gaze estimation, and details experimental results on the MPIIGaze dataset. We give an overview of several approaches which lead to the design of DenseBag and report on their results and shortcomings. We then describe in detail DenseBag. DenseBag consists of multiple Dense Connected Convolutional Neural Networks which are trained independently on bootstrapped samples from the original data and then averaged to give a final result. This reduces the variance and overfitting demonstrated by the individual estimators. The model achieved a Mean Squared Error of 0.00377 on the public test set of the MPIIGaze Dataset (Kaggle). Our code is available at https://github.com/mbbuehler/iPercept.

1 INTRODUCTION

Gaze estimation is a popular research topic in computer vision. Popular approaches are both model- and learning-based. Neural networks have been shown to perform well for the problem of gaze estimation [7]. They recently became the state of the art for the problem of unconstrained gaze estimation [15] [16], where no limitations are enforced on the conditions of observation.

We conduct experiments with several Convolutional Neural Network Architectures and present the performance of several such networks on the task of unconstrained eye gaze estimation. We reason about possible causes for their failure and what lead us to succeed with our final model.

Finally, we present DenseBag, a model which adapts a proven computer vision network architecture, DenseNet [4], to the task of unconstrained gaze estimation and combines it with a bagging technique [1] . The reason for bagging is to reduce variance and overfitting, which we observed to happen for individual models. We show that DenseBag achieves state of the art performance on the MPIIGaze dataset [15] with a mean (angle) validation set error of 5.06 degrees and a final mean squared error of 0.00365 on the Kaggle test set.

2 METHOD

2.1 Arriving at the architecture

To arrive at the final architecture we went through a number of iterations. Initially, inspired by GazeNet [16], we aimed to implement a deep network architecture, VGGNet [12]. However, after training 150 epochs we realized that weight updates were minimal in deeper parts of the network. Training such a deep network from scratch would take much too long. Thus, we decided to pursue two strategies: exploring a range of simpler model architectures, and applying transfer learning (adapting and taking weights from a pre-trained network, in our case VGGNet [2]).

We first implemented a simple convolutional architecture inspired by [8] (2 Convolutional + Max Pooling Layers and 1 Dense Layer). We almost reached the easy baseline. We found that image augmentation (brightness, contrast, Gaussian noise) slightly improved our model (MSE reduced by 0.0005). However, the training error for these models stayed too high. We had high bias.

We decided to use a highly complex model in combination with pre-trained weights. We used the Inception-v3 architecture for feature extraction and tried to fit these features to our numerical output.

We attempted two models, Inception1 and Inception2. Inception 1 had a single dense output layer and Inception four dense layers (with intermittent dropout). In both cases, the pre-trained layers were held fixed. Neither model fit the data sufficiently. We suspect the difference in output domains (1000 class classification versus 2D regression) potentially being too large for any useful transferring of knowledge.

We understood that a good model needed to struck a balance between our (too) simple models and highly complex models (VGGNet, Inception). We went through an iterative process of experimenting with a 3 block convolutional architecture where each block consisted of some convolutional layers, followed by a max pooling layer. We experimented with 1, 2 and 3 layer blocks. The 2 layer network, CCPNet, performed best and beat the easy baseline (Kaggle MSE of 0.00834). We experimented with residual connections [3], image augmentation and trained using different initializations but these did little to improve performance of this model.

2.2 DenseNet Approaches

We finally arrive at an architecture that could fit the training data with a very low MSE (< 0.001): DenseNet. DenseNet was proposed by [4] and takes further the idea of skip connections and connects all convolutional layers to those following them (within what is called a *Block*). The *growth rate* describes how many feature maps are produced by each convolutional layer within a block. Figure 2 demonstrates such a block with a growth rate of k=4. Each block feeds into a *Transition Layer* which performs batch normalization [5], a convolution and average pooling. The final transition layer (which takes the final dense block as input) conducts batch normalization and global average pooling [10]. Its output is then fed into a dense output layer. Before being passed to the first dense block, the input image is passed through a convolutional layer. The architecture in full is visualized in Figure 1.

We take inspiration from an DenseNet implementation on GitHub [9] and adapt it to our needs. In particular we have the following differences to the original DenseNet paper:

- We use an ADAM optimizer [6] where the original paper uses Stochastic Gradient Descent.
- We do not use data augmentation.
- We do not use Dropout [13] layers.

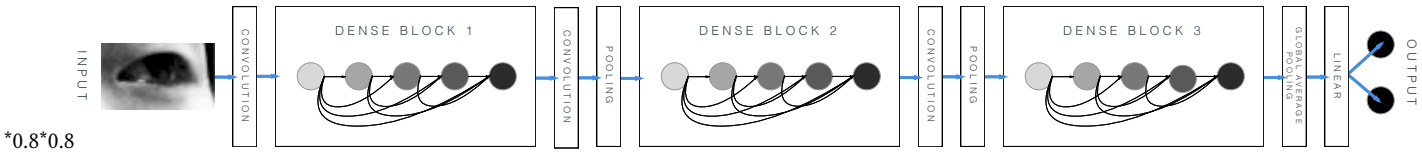

Figure 1: Design of a single DenseNet Architecture within our ensemble.

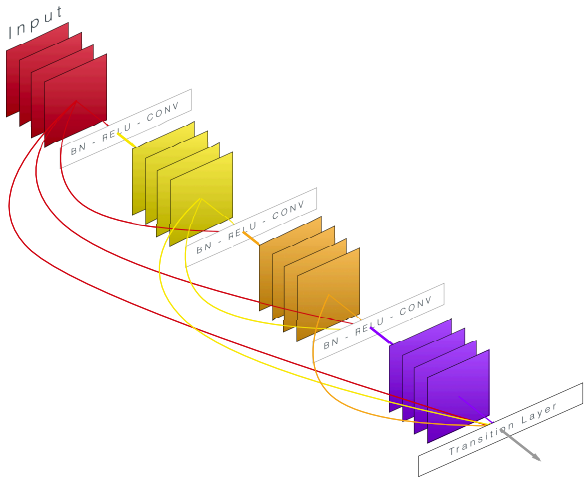

Figure 2: Design of a single 4 layer dense block within our DenseNets. Here the growth rate k = 4. In our implementation k = 12 and each block has 12 layers.

Our DenseNet model starts with a single convolutional layer outputting 16 activation maps. These feature maps are fed into 3 dense blocks. Each dense block has 12 layers, a growth rate of 12 and a single dense layer with output dimension 2. In total, our model has 1,016,754 trainable parameters.

This model is sufficient to beat the hard baseline but overfits the training data. The validation MSE is roughly ten times the training MSE.

In order to lower the variance, we experimented with Dropout [13], L2 regularization[1] and an Elastic Net style cost function[2]. All these methods yielded poor results (i.e. did not reduce the variance significantly).

We also considered to use the head pose angle. However, in [16] they don't find significant improvements for more complex models. Thus we did not use head pose as a feature in the DenseNet. In the following we only use the eye images as input.

2.3 DenseBag

DenseBag solves both the under- and overfitting problem. It uses complex models and applies the technique of Bootstrap Aggregation [1], or Bagging, to alleviate the issues of overfitting outlined in Section 2.2.

DenseBag consists of B single DenseNet estimators, whose predictions are averaged to produce the final estimate.

[1]High error on the training set.
[2]Training no longer converged.

Table 1: Parameters of DenseNet used in the ensemble

Parameter	Value
Batch Size	64
Optimizer	ADAM
Learning Rates	(0.01, 0.001, 0.0001)
Training Epochs	10 + 5 + 5 = 20

Training for a single DenseNet estimator is quick, 35 minutes on average on a Tesla K80 GPU[3]. The low computational cost is a feature of DenseNet described in the original implementation [4] and allows us to proceed with an efficient ensemble method.

In a first version, we averaged the predictions of B models, all initialised with a different random seed. With $B = 9$ models it already achieved the lowest MSE on Kaggle (0.00516). After adding even more estimators to the ensemble we achieved an error of 0.00495 for $B = 45$.

All models in the ensemble were trained using the same input. In order to make the model even more powerful, we trained each model on a bootstrapped sample of the training data. Bootstrapping and combining both the validation and training data to a bigger training set reduced the error further to 0.00381 for $B = 30$.

DenseBag works as follows:

(1) We combine the training (size 30000) and validation (size 2500) sets into one larger dataset (size 32500).
(2) From this dataset we sample a bootstrap sample[4] of the same size as the original data and train a DenseNet architecture from Section 2.2.
(3) We repeat the above step B times, each with a different random seed.
(4) For a test datapoint we average the output of all B networks.

2.3.1 Training. Bagging allows the networks to be trained in parallel. Parameters for the training of one estimator in the ensemble are outlined in Table 1. We split the training into three stages with three separate learning rates. We train for 10 epochs with a Learning Rate of 0.01, 5 epochs with a Learning Rate of 0.001 and 5 epochs with a Learning Rate of 0.0001. As previously stated, this training takes approximately 35 minutes per estimator on a Tesla K80 GPU.

[3]https://www.nvidia.com/en-us/data-center/tesla-k80/
[4]The bootstrap sampling works as follows: We sample with replacement 30000 points from the 30000 training points and 2500 points from the 2500 validation points and combine these into a sample of 32500 points.

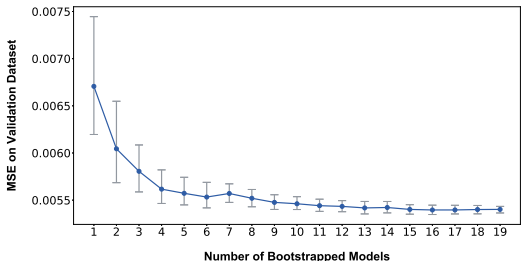

Figure 3: The MSE evaluated on the validation dataset for DenseBag models trained on the training set with varying amounts of bagged estimators B. Point estimates represents the mean of the errors and the error bars represent the 2.5% and 97.5% percentiles of the errors.

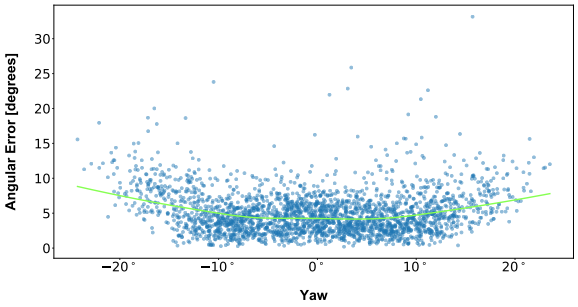

Figure 4: The angular error of a DenseBag model (20 models, trained on the train dataset) as a function of the yaw of samples in the validation set.

3 RESULTS

In order to get a sense for our model's accuracy, we trained 20 bootstrapped base models on the training dataset and evaluated on the validation dataset. To examine how the number of base estimators chosen, B, affected results we did the following:

- for each B in 1 to 19 we calculated the error on the validation set of all possible ensembles[5]of size B made up from the 20 base models.
- for each B we took the mean and 2.5% and 97.5% quantiles of these errors.

The results of this experiment are outlined in Figure 3. We see that the MSE reduces as we increase B and that the variance of the bagged estimator also decreases.

Figure 5 shows a similar curve for DenseBag models trained on the combination of the train and validation datasets (see Section 2.3) and evaluated on the Kaggle dataset. Here we also see that after 30 base models, improvements in score are minimal. Our final model uses 99 base DenseNet models and achieved a score of 0.00365.

Finally, Figure 4 demonstrates a problem outlined in [16] that also presents itself in our model. Due to the lack of data in wider gaze ranges, the model is inaccurate for wider larger yaw values.

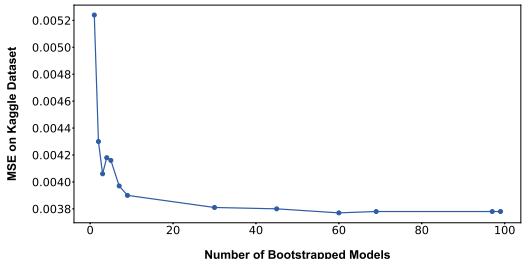

Figure 5: The MSE evaluated on the Kaggle dataset for DenseBag models with varying amounts of bagged estimators B.

4 DISCUSSION

A single instance of our DenseNet model has 40 layers. Despite this considerably large number, its 1,016,754 parameters are optimised in less than 35 minutes on average on a Tesla K80 GPU. This supports the claim that skip connections in models of medium complexity reduces training time. However, in contrast to [4], we find a single DenseNet to suffer from high variance. After training for 20 epochs, the MSE of a single model on the training set is 10 times lower than the MSE on the validation set. This is a clear indicator for high variance and overfitting.

Ensemble methods, such as bagging or boosting, are classical approaches to reduce the variance of estimators by keeping the bias constant. Thanks to the efficient training of densely connected networks, it is computationally feasible to apply these techniques to deep neural networks. In our case, the problem of eye gaze estimation, we showed that it is sufficient to train 30 estimators in order to achieve state of the art performance. This number could potentially be reduced by training densely connected networks that overfit less. An option to this could be adding bottleneck layers, compression [4] or a combination of both. Furthermore, tuning hyperparameters and using other optimisation techniques could further reduce the time for single estimators to converge.

[5]There are $\binom{20}{B}$ such possibilities. Due to computational cost we cap at 100 configurations for those in excess.

5 CONCLUSION & FUTURE WORK

This paper presents a high performing, efficiently trainable model for the task of unconstrained eye gaze estimation. It also demonstrates the power of bagging as a method to reduce overfitting.

Several avenues lay open for exploration for further work. Using bottleneck layers and compression as proposed by [4] could reduce variance in single DenseNet estimators.

Another technique which is popular in the domain is the use of synthetic input data [14], due to its efficiency of generation, data variety and precision labeling. Approaches of using networks to refine synthetic images [11] could provide useful input data which could prevent overfitting and lead to even better results.

ACKNOWLEDGMENTS

We would like to thank the ETH AIT Lab[6] for organizing this challenge and writing the skeleton code. Many thanks to Microsoft Azure[7] for providing us with the resources to train DenseBag.

REFERENCES

[1] Leo Breiman. 1996. Bagging predictors. *Machine learning* 24, 2 (1996), 123–140.
[2] Davi Frossard. 2016. VGG in TensorFlow. http://www.cs.toronto.edu/~frossard/post/vgg16/
[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385
[4] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. *CoRR* abs/1608.06993 (2016). arXiv:1608.06993 http://arxiv.org/abs/1608.06993
[5] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). arXiv:1502.03167 http://arxiv.org/abs/1502.03167
[6] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014). arXiv:1412.6980 http://arxiv.org/abs/1412.6980
[7] Kyle Krafka, Aditya Khosla, Petr Kellnhofer, Harini Kannan, Suchendra Bhandarkar, Wojciech Matusik, and Antonio Torralba. 2016. Eye Tracking for Everyone. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[9] Yixuan Li. 2018. DenseNet-tensorflow. https://github.com/YixuanLi/densenet-tensorflow
[10] Min Lin, Qiang Chen, and Shuicheng Yan. 2013. Network In Network. *CoRR* abs/1312.4400 (2013). arXiv:1312.4400 http://arxiv.org/abs/1312.4400
[11] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russell Webb. 2016. Learning from Simulated and Unsupervised Images through Adversarial Training. *CoRR* abs/1612.07828 (2016). arXiv:1612.07828 http://arxiv.org/abs/1612.07828
[12] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).
[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[14] Erroll Wood, Tadas Baltrušaitis, Louis-Philippe Morency, Peter Robinson, and Andreas Bulling. 2016. Learning an appearance-based gaze estimator from one million synthesised images. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*. ACM, 131–138.
[15] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2015. Appearance-Based Gaze Estimation in the Wild. *CoRR* abs/1504.02863 (2015). arXiv:1504.02863 http://arxiv.org/abs/1504.02863
[16] Xucong Zhang, Yusuke Sugano, Mario Fritz, and Andreas Bulling. 2017. MPIIGaze: Real-World Dataset and Deep Appearance-Based Gaze Estimation. *CoRR* abs/1711.09017 (2017). arXiv:1711.09017 http://arxiv.org/abs/1711.09017

[6] https://ait.ethz.ch/
[7] https://azure.microsoft.com/en-gb/