DD2424 - Project Report: Classifying Food with WideNet

Group 16: Felix Gudéhn, Mazen Mardini, Theodor Uhmeier

Abstract

In this project the authors adapt the methodology from the paper on Wide Residual Networks [6] to a large collection of food images with the intent to train a classifier on this dataset. The intended goal of the project is to explore what performance can be achieved on this much larger and messier real world data set. The final result of the best performing model was 67% top-1 accuracy on the testing dataset, which outperforms the both the random-forest and CNN used in the reference paper from which dataset was obtained [1].

1 Introduction

This project is based on the first basic project suggestions (Explore training ConvNets from scratch for image classification), with this we will focus on the problem of image classification through the use of deep convolutional neural networks. Specifically we will attempt to apply the WideResNet [6] model to an image dataset of different food types [1]. The end goal of the project will be to produce a model being able to classify these different foods with sufficient accuracy.

The domains of computer vision and image classification have seen massive improvements in accuracy over the last decade. Most of these improvements have been through the use of CNNs, with a flurry of papers being published on the topic [1, 2, 3, 4, 6]. For this reason, we decided to take on a problem space which could find solutions related to convolutional neural networks.

Classifying food is a complex domain to try to solve due to the high variability in appearance of different foods. However, humans are able to solve this task sufficiently well which makes it an interesting topic of research. Being able to classify food could allow for automated solutions of calorie counting and generating probable lists of ingredients for a dish which could all be useful applications of a solution in this domain.

2 Related Work

As previously mentioned, CNNs have been used for image classification successfully and have seen gradual improvements in recent years with some notable steps on the way being AlexNet, VGG and ResNet. In this paper we are focusing on ResNet [2], and more precisely WideResNet [6]. To understand the idea of the WideResNet we first have to explore the key concept of ResNets and how they work.

The idea of Residual connections in CNNs was first introduced in [2]. The motivation behind it stems from the problem of deepening of networks sometimes having both worse training and testing performance. As this problem affects both training and testing it is not overfitting and thus it does not imply the issue being overly complex model in the self. The solution for this problem is based on the idea that a network to which you add layers with weights representing an identity function will never degrade the performance but rather keep it the same. So instead of having the untrained weights of added layers be close to the zero as they normally would with they start with the baseline of added layers starting close to weights for the identity function, and learning from there. To implement these "shortcuts" or so called Residual Connections where the input of one block(multiple conv-layers)

would be kept and added to the output at the end of the block to only have the added layers make small changes to the output instead of large ones. The result of this was the ability to create deeper networks and still see performance improvements.

The idea of creating wider, rather than deeper, residual networks was first proposed and implemented in the paper by Zagoruyko and Komodakis [6] in 2017. The authors explored the effects of increasing the width of subsequent layers and contrasted them against the results of making the network deeper. Wider networks were proven to achieve similar accuracy with less parameters than there deeper counterparts, thus improving efficiency and reducing training times.

3 Data

For our project we used a labelled dataset of 101,000 images of various types of food such as 'apple pie', 'carrot cake' and 'caesar salad', with 101 different classes in total. The dataset was compiled as part of a paper titled 'Mining Discriminative Components with Random Forests' where the authors authors attempted to train a specialized Random Forest classifier [1].

The images in the dataset have all been taken under different conditions since the source of the images was a web application that allowed users to upload images of food. This, together with the inherent variability of food, results in each individual class to containing a lot of noise [1]. In Figure 1 we show some of this variability within the class 'samosa' where each image has different lighting, orientation and color.

Figure 1: Some randomly sampled images from the 'samosa' class, displaying the variation between images in the same class.

4 Methods

The way we went about creating our WideResNet solution was by taking inspiration from existing implementations and architectures. We relied heavily on PyTorch Vision's ResNet code as well as the code written by the author of the WideResNet paper [5] and the architecture that was used there. This way we feel confident that we are using established methodology, backed by the original WideResNet paper and implementations from PyTorch developers.

We began by building our own model-class based on the architecture in the torchvision.models.resnet module. Initially, our version ended up having the exact same architecture, albeit being significantly shorter. The class BasicBlock, representing ResNet-blocks, from the module was reused, and so were a couple of other functions.

Although we had implemented a WideResNet, it had some architectural differences to the one implemented by the author of the original paper. On such difference was, citing from the WideResNet paper:

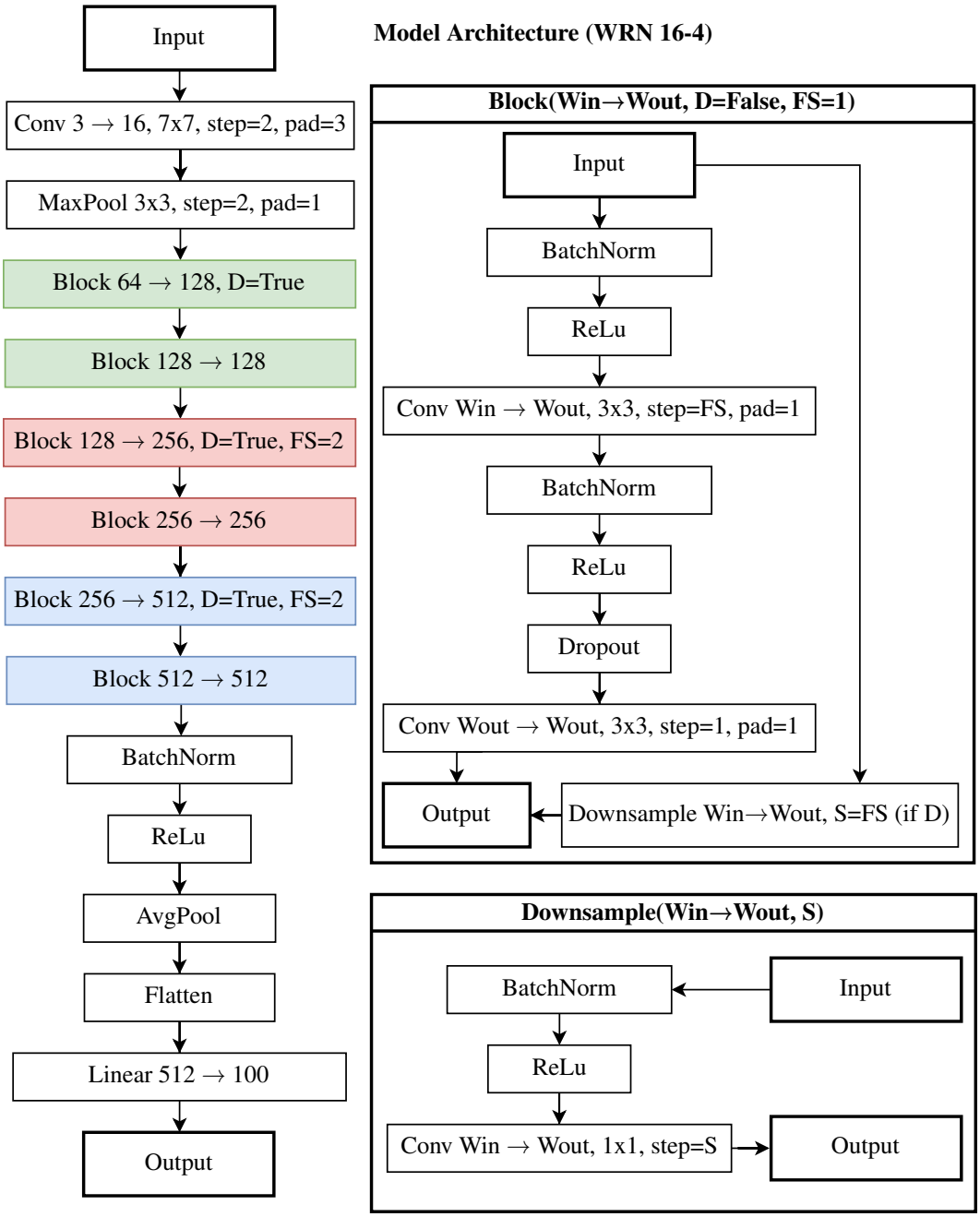
"Compared to the original architecture [11] in [13] the order of batch normalization, activation and convolution in residual block was changed from conv-BN-ReLU to BN-ReLU-conv. As the latter was shown to train faster and achieve better results we don't consider the original version." [6]

Incorporating these changes from the WideResNet paper into our model required us to have our modified version of BasicBlock in the final version.

Another difference was a down-sampling (max-pooling) step early in the model, this was not part of the paper architecture. However, we kept it because training without this step would end up being about 3-4 times slower.

The final architecture can be seen in Figure 2.

Figure 2: Our final model architecture using width=16 and depth=4 (in short: WRN 16-4).



4.1 Collecting experimental data

It's important to retain data, such as test accuracy, running time, and average loss for each epoch to be able to visualize the training process. To do this reliably we made our program save a JSON entry to a file for each (successful) epoch. We also saved the last trained model as well as a copy of the model with the highest test accuracy. The program also generated plots automatically for convenience.

4.2 Training

In order to ensure that our models are properly applying regularisation and not overfitting too easily on the dataset, we opted to first use smaller image sizes for our training. Once we were sure that our models were able to properly learn the underlying distribution, we then proceeded to increase batch and image sizes to match the capabilities of the GPU on GCP and Google Collab.

5 Experiments

Our initial experiments included runs where we tried to train the Wide Residual network on 32x32, 64x64 and 128x128 size images with 3 color channels. The best performing model achieved a test accuracy of 67.5% after about 80 training epochs using an input image size of 128 by 128 pixels. As in the paper on the original implementation of WideResNet we divided the learning rate by 10 after a fixed number of epochs(40 and 60) in order to refine and not overshoot the optimal maxima. As seen in Figure 3, the reduced learning rate led to a jump in accuracy after about half of the training time. The second drop in learning rate does however not seem to have had the same effect.

Comparing to the original creators of the dataset who were able to achieve a test accuracy of 50.76% with a Random Forest and 56.40% with a CNN[1] our test accuracy of 67.5% is able to outperform both of those methods.

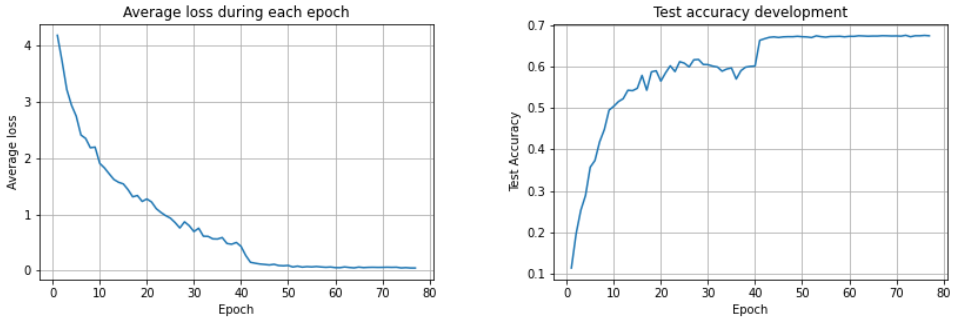


Figure 3: Left: Average loss over training of the best performing model. Right: Test accuracy of best performing model over epochs.

We examine the final classification accuracy for each of the 101 food classes and found that certain classes such as 'edamame', 'pho' and 'oysters' were easier for the model to classify. Conversely we also found that there were certain classes like 'apple pie', 'foie gras' and 'steak' where the was able to predict the correct class less than 50% of the time. The top five and bottom five class prediction accuracies can be seen in Figure 4.

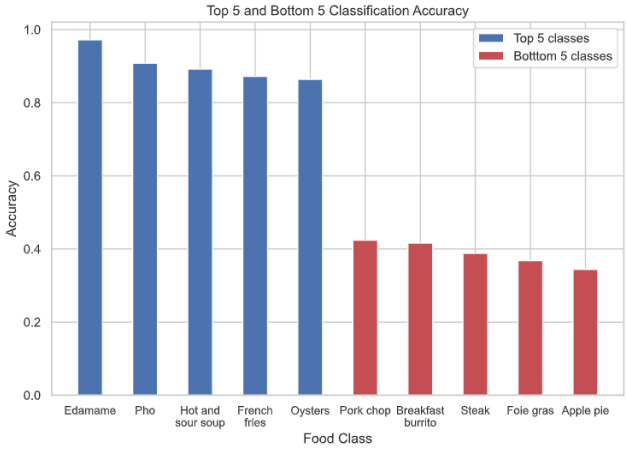


Figure 4: The top 5 and bottom 5 class prediction accuracies on the dataset. Edamame beans and apple pie were the easiest and hardest to classify respectively.

In order to understand what the model is learning we decided to visualise the filters within the convolutional layers of the model (see Figure 5). The input image is passed through each of the convolutional layers and the result is then displayed in image format. We observe that deeper layers contain lower level abstractions than the earlier ones.

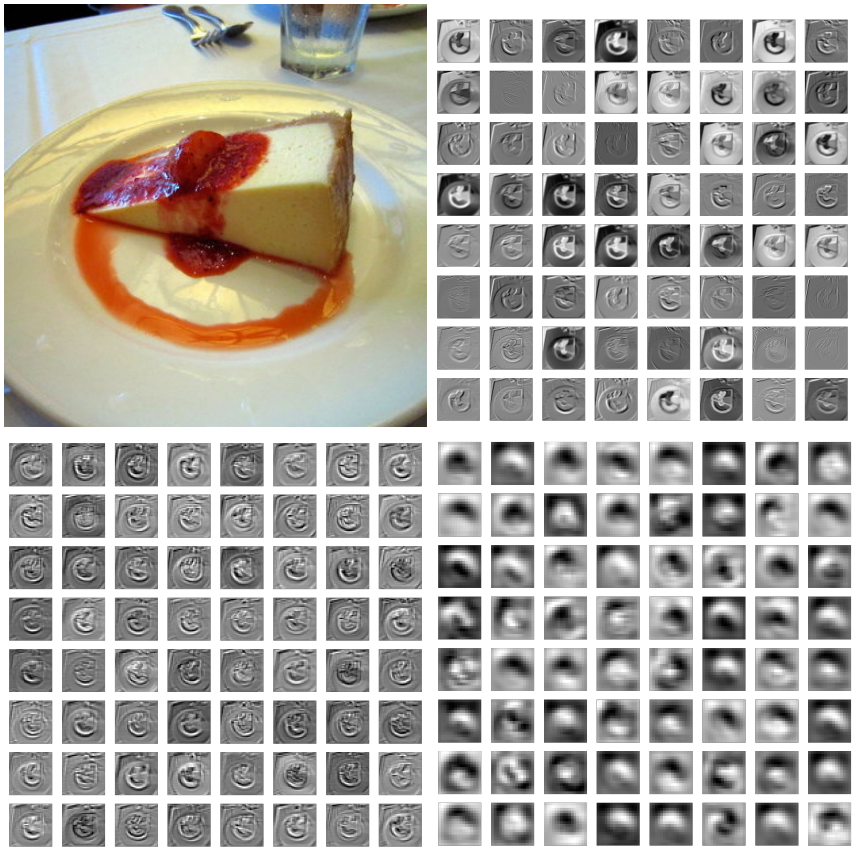


Figure 5: Visualisation of the filters of 3 of the 16 layers of the model. Top: input image and first layer filters. Bottom: filters from layers 9 and 16. The input image has the label 'cheesecake'.

6 Conclusion

Classifying food from images taken under different conditions is a complex problem due to the variability within categories. Our results show that a Wide Residual Network was able to outperform the CNN's results in one of the references papers working on the same dataset.

Certain classes were easier for the classifier to predict, specifically edamame beans, pho and hot and sour soup. The hardest to predict classes were dishes such as steak, foie gras and apple pie. The highest scoring classes all have very little noise and variation whereas the hard to predict classes have a lot of variation which explains to some extent why these classes were harder for the model to predict.

Acknowledgments and Disclosure of Funding

We would like to thank the DD2424 course responsible for providing us the resources necessary to run models on the Google Cloud Platform. Furthermore we would like to acknowledge the authors of the paper who made the food dataset publicly available.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – Mining Discriminative Components with Random Forests. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, volume 8694, pages 446–461. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv:1512.03385 [cs]*, December 2015. arXiv: 1512.03385.

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

[4] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, April 2015. arXiv: 1409.1556.

[5] Sergey Zagoruyko. Wide residual networks. `https://github.com/szagoruyko/wide-residual-networks`, 2016.

[6] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *arXiv:1605.07146 [cs]*, June 2017. arXiv: 1605.07146.