

COMP9444 Neural Networks

Assignment 2

Charlie Wyatt

Adam Schuman

# Architecture

A ResNet34 was used, some of the architecture was taken from the following git repository.(1). ResNet was chosen since it's a well documented architecture that has seen success in lots of past image classification problems, including surpassing VGG and AlexNet since the skip connection allows it to train deeper without exploding or vanishing gradients (<https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96> and <https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception> /). We tested an 18 layer ResNet, but ultimately decided upon a 34 layer ResNet since the added complexity improved the test accuracy without overfitting the data and fit beneath the 50 Mb limit (47.3 Mb).

# Loss Function

Cross entropy was the loss function used, a standard loss function for multiclass classification.

# Optimiser

An SGD optimiser was chosen. Our research showed that SGD optimizers are commonly used for Resnet’s and that momentum improves them significantly. (2) An Adam optimiser was also tested but it had a lower accuracy.

# Metaparameters

## Batch Size

A small batch size of 4 was chosen. A power of 2 was chosen for memory efficiency and we found that smaller batch sizes stopped the gradient descent from becoming stuck in a local minima. Of the batch sizes tested (4, 8, 16, 200 and 248), 4 was the best value.

## Learning Rate Scheduler

An exponential learning rate scheduler was implemented but it shrunk out learning rate to zero at approximately 50 epochs and the model would no longer train. We removed the learning rate scheduler, and it improved the accuracy.

## Weight decay

We had some overfitting problems and so added a small weight decay to compensate.

## Learning rate

Found through research and experimentation

# Transforms

Due to such a limited training set, we increased our training data through data augmentation. For this data augmentation, only training images were transformed since the testing images should represent the raw input data.

Given the training data, not all tranforms are appropriate. Colour changes are not appropriate since it is a primary discriminator for determining breed. Cropping was also not used since the cat could be cropped out of the picture. RandomAffine and RandomPerspective were not used since they negatively impacted results although the reason is not fully understood.

Lastly, we tried normalizing both testing and training datasets, however, this made our model perform poorly, although the reason why is not clearly understood.

The transformations that were applied were rotation, horizontal/vertical flipping, sharpness changes and colour “jittering” since these changes don’t negatively affect the modelling and all improved results.

# Dropout

Although not usually used for convolutional networks (3), dropout was added to reduce overfitting. Before applying dropout, the training accuracy was exceeding 85%, with a testing accuracy of 75%. With dropout, the testing accuracy rose slightly to 77%.

# Results

Figure 1. Confusion matrix and accuracy of neural network

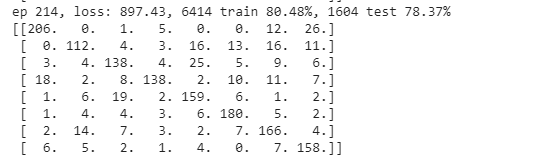


Figure 1 shows the confusion matrix and results for training and testing of the network. The final accuracy plateaued around 78%, at around 300 epochs. Evidently, some overfitting occurred, but this was reduced from the initial value with the aforementioned techniques.

# References

1. <https://gist.github.com/nikogamulin/7774e0e3988305a78fd73e1c4364aded>
2. <https://openreview.net/pdf?id=B1Yy1BxCZ>
3. <https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html>

Rationality for our decisions -

TRANSFORMS-

Due to such a limited training set, increasing our training data through data augmentation was necessary.

For this data augementation, only training images were transformed since the testing images

should represent the raw input data.

Transforms we used-

Only some transformations are appropriate to the training data since cat breeds are not

recognisably different depending on some changes to the image, e.g. whether or not it is flipped.

Using this logic, the appropriate transforms were rotation, horizontal/vertical flipping, sharpness

changes and brightness "jittering", since these do not change the appearance of

the cat in any meaningful way to the model.

Transforms we did not use

Some transformations which did introduce issues are hue and colour changes,

since identify the breed of a cat is largely dependent on its colour. Cropping was not used

since there is a chance the cat could be cropped out of the picture. RandomAffine

and randomPerspective both made the model worse, the reason for this is currently not

understood. Lastly, we tried transforming with normalization in both testing and

training datasets, however, this made our model perform poorly, although the reason why

is not clearly understood.

LOSS\_FUNCTION-

Cross entropy is a standard loss function for multiclass classification

OPTIMIZER

SGD is usually the best with Resnet. Momentum apparently plays a large part in training a ResNet.

https://openreview.net/pdf?id=B1Yy1BxCZ

BATCH SIZE

Small batches make gradient descent faster (https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e)

GOOD LINK!: (https://stats.stackexchange.com/questions/164876/what-is-the-trade-off-between-batch-size-and-number-of-iterations-to-train-a-neu)

Tried a batch size of 8, 16, 200 and 248. A batch size of 4 increased the testing

accuracy the most and stopped it from getting stuck in small local minima during gradient descent.

With a batch size of 248, the model would occasionally get stuck around 50% testing accuracy

LEARNING RATE

Found through research and trial and error

LEARNING RATE SCHEDULER

No scheduler. We introduced a gamma (? can't remember if this was right) scheduler but this shrank our gradient to zero.

After about 50 epochs, our model's loss would not move.

WEIGHT DECAY -

Some issues with overfitting, added a small weight decay to try to correct for this since

weight decay places a small penalty on the weights of the model.

DROPOUT

Although dropout is not usually used for convolutional networks, (https://www.kdnuggets.com/2018/09/dropout-convolutional-networks.html)

we added it between our convolutional layers and our last step and found that it reduced our overfitting problem slightly.

Before dropout:

When testing accuracy got to 75%, training accuracy increased to about 85%

but testing accuracy remained relatively similar. This is a sign of overfitting

After dropout:

Still began to overfit at 75%. Reached 75% testing accuracy around the 100th Epoch.

As training continued the training accuracy improved much faster than the testing accuracy,

however testing accuracy reached a peak accuracy of 79%, indicating that the model

with dropout had less overfitting issues than the model without dropout.