Project 1 - Cat & Dog Classification

Henrik Daniel Christensen[*hench13@student.sdu.dk*]
Frode Engtoft Johansen[*fjoha21@student.sdu.dk*]

DM873: Deep Learning
University of Southern Denmark, SDU
Department of Mathematics and Computer Science

1 Introduction

The objective of this project is to develop a deep learning model capable of distinguishing between images of cats and dogs. The task involves training a neural network using a dataset containing 3,600 images, equally divided between the two categories.

2 Explorative Analysis

The dataset contains a mix of different breeds of cats and dogs. The images are very diverse in terms of size, orientation, lighting, focus, and resolution. Also, the images are taken from different angles and distances. The images are mostly around 200-500 pixels width/height and are all in color, meaning they have 3 color channels. In Figure 1, examples of cats and dogs from the dataset are shown.

(a) Cats **(b)** Dogs

Fig. 1. Sample images of cats and dogs from the dataset.

From the images, we see that cats and dogs have different features that can be used to distinguish between them. For example, cats have generally shorter faces often with triangular ears and pointed noses, while dogs generally have longer snouts and have more different ear shapes. Moreover, the eyes of cats are generally more sharp-shaped, while dogs have rounder eyes. The goal is therefore to create a model that can learn these features.

3 Base Model

First, we set out to develop a base model. However, first, we had to decide which image size to use. We chose an image size of 224 by 224, since almost all images are bigger on each axis, so we almost only shrink pictures. The pretrained models also uses resizing to 224 pixels making our model more comparable to the pretrained ones. Resizing the images has the effect of reducing the size of the input to the model which will reduce the accuracy a little bit, but make it train significantly faster.

Next, we had to decide how many layers our model should have. As the task is relatively simple, we decided to use 4 convolutional layers and 3 fully connected layers. Since the input images are in color and 224 pixels on each axis, inputting that image directly into the fully connected layers would result in a 224*224*3 = 150528 input size into the model. We would like to reduce this further, so the fully connected layers does not get an input of 150528. The convolution layers extract features, which makes it possible for the neural network to recognise a specific feature no matter where it is in the image, instead of having to learn that feature for every possible position in an image.

We use cross entropy loss as our loss function since this loss function is a popular and good loss function for classification tasks. We use adam as the optimizer since it combines the strength of several other optimizers.

The base model is shown in Table 1.

	Output kernels/features	Kernel	MaxPooling	Activation
Conv2D w/	32	3x3	2x2	ReLU
Conv2D w/	64	3x3	2x2	ReLU
Conv2D w/	128	3x3	2x2	ReLU
Conv2D w/	256	3x3	2x2	ReLU
Linear w/	256	-	-	ReLU
Linear w/	128	-	-	ReLU
Linear w/	2	-	-	-

Table 1. Base Model.

Results of the base model after 25 epochs are shown in Figure 2.

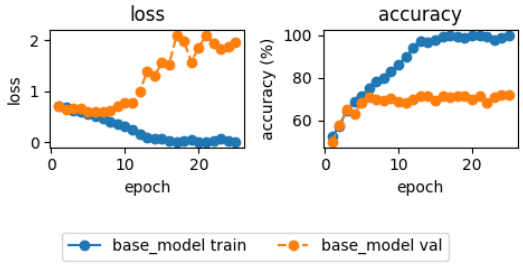


Fig. 2. Base Model Results.

We chose to train the model in 100 epochs. This is a sufficiently high number that the model gets to be properly trained, but still a reasonable number to keep within a reasonable time to train the model. More epochs is good until a certain point when the model starts overfitting. As an improvement we could have included some early stopping in case the model began overfitting. We chose a learning rate of 0.001. A learning rate that is too small could result in a very slow learning process that might get stuck, and a too big learning rate could result in learning a suboptimal set of weights or an unstable learning process. A smaller learning rate requires more epochs for a better result. The default learning rate is 0.01. We chose a smaller learning rate since we wanted our learning process to be very stable, and since we run 100 epochs it seemed appropriate.

Clearly, the base model is overfitting, one way, especially for small datasets as in this case, to reduce overfitting is to use data augmentation, enabling the model to learn from more data.

4 Regularization

4.1 Data Augmentation

We chose to augment the data in a few different ways since we want to generalize the data so the important features remain. Using augmentation ensures that it will get better accuracy when faced with new images. In other words, it reduces overfitting.

The images are of varying quality and taken in different lighting, and this is also what we can expect from the final test set. For this reason we change the brightness, saturation, and contrast. Furthermore, the pictures are taken from different angles so we rotate the pictures slightly and flip them horizontally. Generally, images of animals are not vertically flipped, so we chose not to do this, but of course, some cases may exist.

The pictures are also cropped differently to get finer details from the images, as well as the images becomes scale invariant.

Even though some of the pictures are more blurry than others, we chose to not use blurring as data augmentation, as it effectively just reduces the data of the image without reducing input to the model.

We shear and translate the pictures to emulate pictures taken from different angles of the animal.

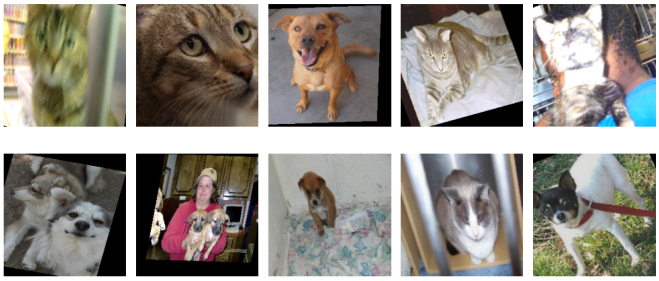
Since the training dataset is quite small, we wanted to create more data by reusing the same images but with random augmentations, but in the end we found out that our results did not improve from this.

Many different data augmentation techniques exists, to find the best data augmentation techniques for this task, one image was used to test different techniques.

The techniques used for this task as well as augmented sample images by these techniques are shown in Figure 3.

Data Augmentation	Parameters
RandomHorizontalFlip	50%
ColorJitter	brightness=0.2 contrast=0.2 saturation=0.2 hue=0
RandomAffine	degrees=25 translate=(0.1, 0.1) scale=(0.7, 1.3) shear=(-10, 10)

(a) Data Augmentation Techniques.



(b) Sample images after augmentation.

Fig. 3. Data augmentation techniques.

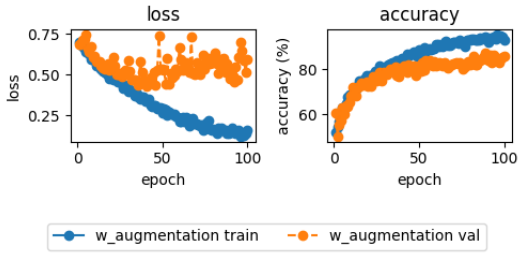
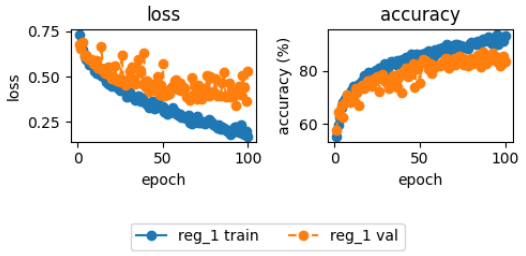


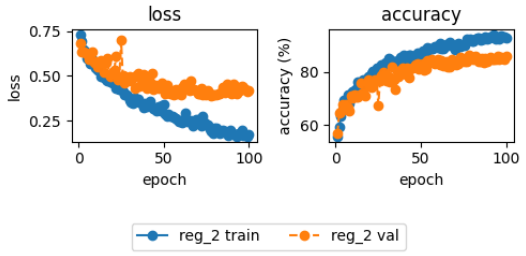
Fig. 4. Sample images after augmentation.

4.2 More Regularization Techniques

1: - 93- 84
2: - 93- 85
- loss mere stabil på 2. Den er med weight decay og halvering af learning rate hver 25 epoker. - dropout layer



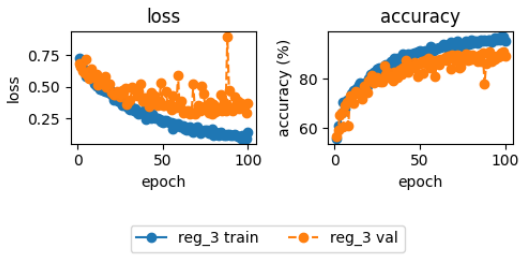
(a) Regularized model 1.



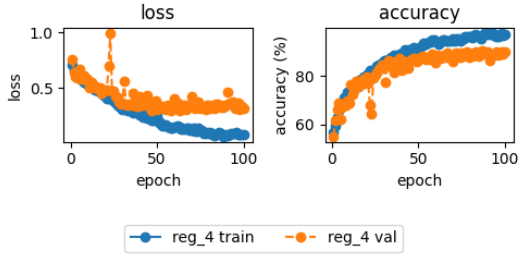
(b) Regularized model 2.

Fig. 5. Results using regularization.

4.3 Adding one more convolutional layer



(a) Regularized model 3.



(b) Regularized model 4.

Fig. 6. Results using one more convolutional layer.

5 Prediction

For prediction on the test set, we choose the model

6 Pretrained Model

To see if we could improve the model, a pretrained AlexNet model is implemented. This model is pretrained on the ImageNet dataset, which contains 1.2 million images and 1000 classes, also including cats and dogs. However, to use the pretrained model, the output layer is modified to fit the binary classification task of cats and dogs.

The results of the pretrained model after 25 epochs are shown in Figure **??**.

7 Conclusion

We ended up with a model that has ?? results. The model we used was ?? with ?? layers having XX parameters. This result is pretty good, but it could be improved. It is worth noticing that some of the pictures are very hard to classify, like dog.1335 and dog.1395.

- compare results to pretrained model - why is the pretrained model better?

The model could have been improved in various ways. First of all more training data would very likely improve the model a lot. In some of the pictures its very hard to see whether its a cat or a dog and its doubtful that the model would learn anything from the picture, so removing those pictures from the dataset would potentially help the model learn more meaningful features and also train faster. We could have run epochs until the model began overfitting and then stopped at that point. If time was not a factor this is probably what we would have done. We could have also experimented more with the different parameters like learning rate and weight decay. We could have tried even more data augmentations to see what works best. We only rotate the pictures up to 25 degrees, which means that the model as an example would have a hard time recognising pictures of animals upside down.

7.1 Individual Contributions

A Code

hej

B Notebook

hej