Exploring Neural Networks: Classifiers and Generative Models

Abstract

In this report we attempted to build an image recognition classifier for the dataset CIFAR-100 and to generate an image of a pegasus (a horse with wings) from CIFAR-10 dataset using a generative model. Our classifier model reached an accuracy of 49%, however a lot more work could be done on this. The generated pegasus image is satisfying, although some further fine-tuning is recommended.

1 Introduction

Today image recognition is a rapidly growing and developing field of computer science, thanks to the various machine learning and AI techniques and technologies. Several training datasets are available online for this purpose, however in this study our focus was to implement a code for image classification on the CIFAR-100 dataset and to generate an image of a Pegasus using Adversarially Constrained Autoencoder Interpolation of the latent spaces of bird and horse images from the CIFAR-10 dataset.

2 The classifier

2.1 Our model

To train the classifier we first looked up and studied several successful implementations for classifiers such as Alexnet (older), Inception-v1, v2, v3, v4 (GoogLe Net), Residual Networks, Inception-Resnet v1, v2. Having tried to implement these models we quickly realised that they are too complicated for our scenario, since most of them were designed for ImageNet dataset which has much larger pixel dimension than our 32x32 images. Thus the idea was to follow the structure of these methods but optimise them to the CIFAR dataset.

At first we did not manage to work out an efficient and accurate model following the state-of-the-art methods as many of them increased in accuracy very slowly and hardly ever reached 40%. In fact, we achieved faster and more accurate results using simply a few convolutional layer, pooling layer, batch normalisation and ReLU. That is, with this simple method (similar to Alexnet) we achieved around 40% only in a few minutes (epoch = 10).

On the other hand, to avoid the vanishing gradient problem and to be able to further expand our network (to go deeper) we chose to implement a *resnet*-type model, different than the ones in the studied papers. The idea was to apply 2-2 convolutional layers in both *blockA* and *blockB* residual blocks, and repeated each block 3 times summing them up with the previous outcomes. After *blockA* we applied a reduction module to reduce the size of the images using $stride = 2$ and increased (doubled) the number of feature maps, and again fed this into the the sequence of *blockB*. Note that since the size of the image was already relatively small (32x32), it was tricky to do any kind of reduction/pooling in the *stem* part of the model since it could easily lead to loss of edges. Thus in the *stem* part (before the residual blocks) we only applied a $1x1$ convolution on the original image to generate feature maps of the same size. At the end of the second residual block a max pooling layer was applied to further reduce the spatial dimensions of the images before applying the final fully connected layer. See Figure 1 for visual representation.

2.2 Results

Using the model above, we managed to achieve around 49% accurate results on the test data using 50 epoch, however on 10 epochs simpler implementations outperforms our resnet model.

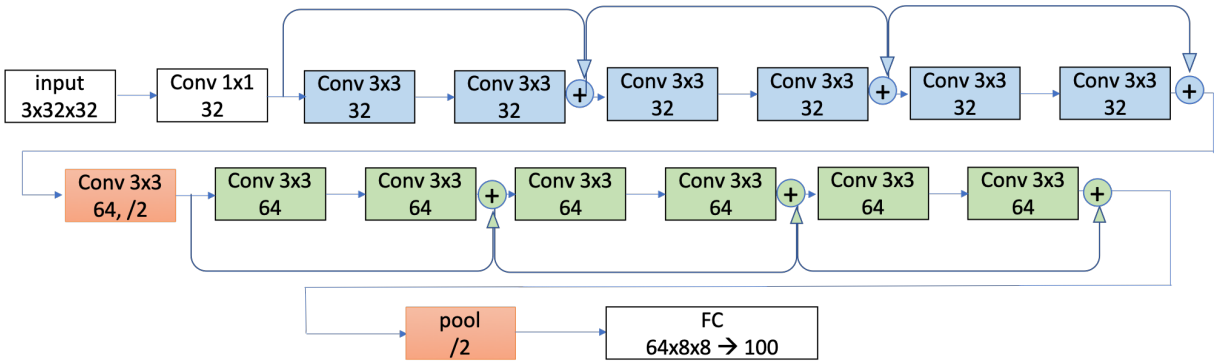
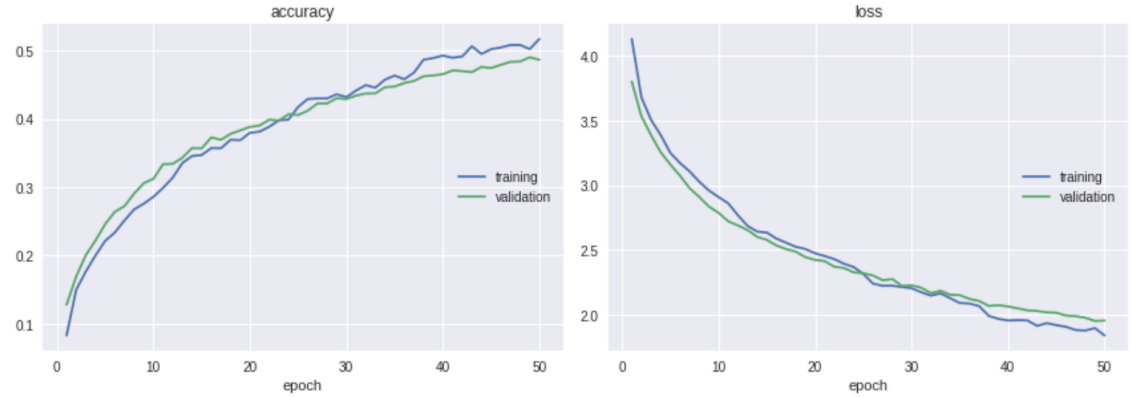


Figure 1: Our implementation of the residual network for the CIFAR-100 dataset. *BlockA* seen in blue, *blockB* in green, and the orange layers indicate the reduction modules/pooling.

On the other hand one could observe the vanishing gradient problem on 50 epochs for simple models, whereas our implementation kept gradually increasing in accuracy. For exact outcomes see Figure 2.




```
accuracy:
training   (min:    0.083, max:    0.517, cur:    0.517)
validation (min:    0.128, max:    0.490, cur:    0.486)

loss:
training   (min:    1.843, max:    4.126, cur:    1.843)
validation (min:    1.955, max:    3.798, cur:    1.958)
```

Figure 2: The accuracy and loss of our implemented model using 50 epochs. The highest test (validation) accuracy is 49%.

3 Generating a Pegasus

Generating a new, unseen image is possible using generative networks and some training dataset. Moreover, it is also possible to create an image based on not only 1 but 2 different *kinds* of image using interpolation. In this study we will apply a linear transformation to the latent space of the encoded images and decode the resulting latent space back into the *generated* image. First we need an encoder and decoder function where the decoder does the inverse transformation in reversed order in comparison to the encoder. After some trial and error investigation we derived our own implementation for the autoencoder, which is shown below on Figure 3. To interpolate the latent spaces, we applied a simple linear transformation as follows:

$$pegasus = \alpha \cdot bird + (1 - \alpha) \cdot horse, \tag{1}$$

where α is the probability of the bird latent space, and thus $\alpha - 1$ is the probability of the horse latent space. Note that *pegasus* above still needs to be decoded.

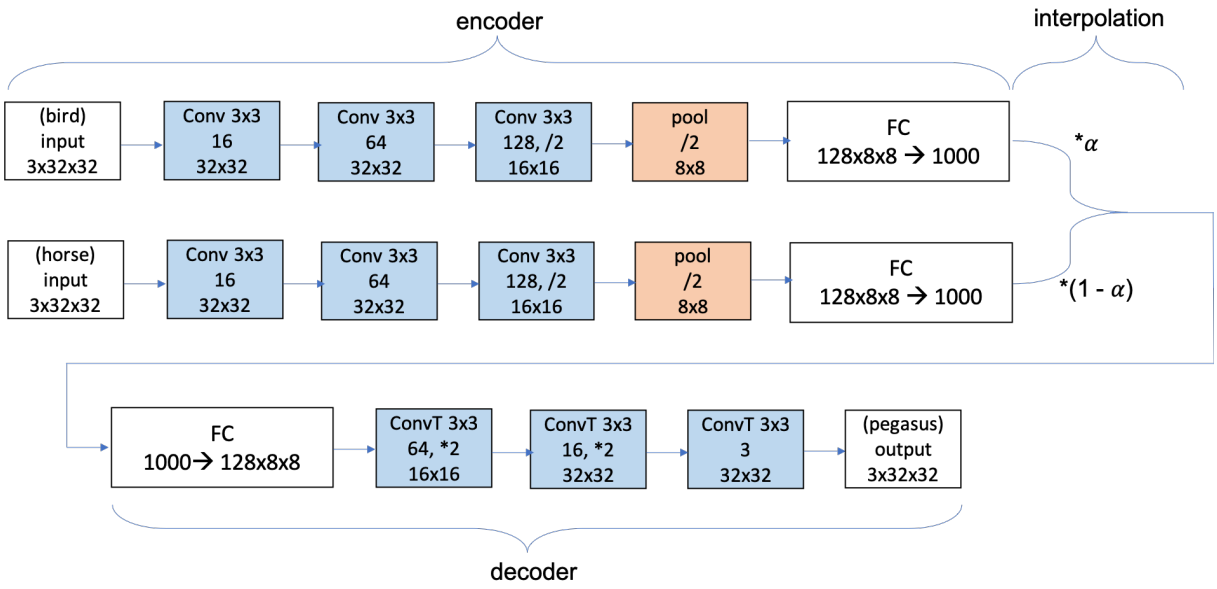


Figure 3: The generative model (autoencoder) in our implementation for generating a pegasus. ConvT stands for Transpose Convolution layer, and α is probability.

Figure 4: Generative horse (top left) and bird (top right) and their interpolation (bottom).

On Figure 4 one can observe the generated horse and bird and their interpolation forming the pegasus. Notice the difference between the bottom two images. They have been generated using the exact same parameters, only the training happened separately, which explains the difference. After looking into this, we could say the left one is the general outcome, and the right one only happens rarely.

After running some tests we found that $\alpha = 0.4$ for bird and hence 0.6 for horse is a sensible choice overall, however one could continue fine-tuning this. Using 20 epochs the model achieved a low training loss:

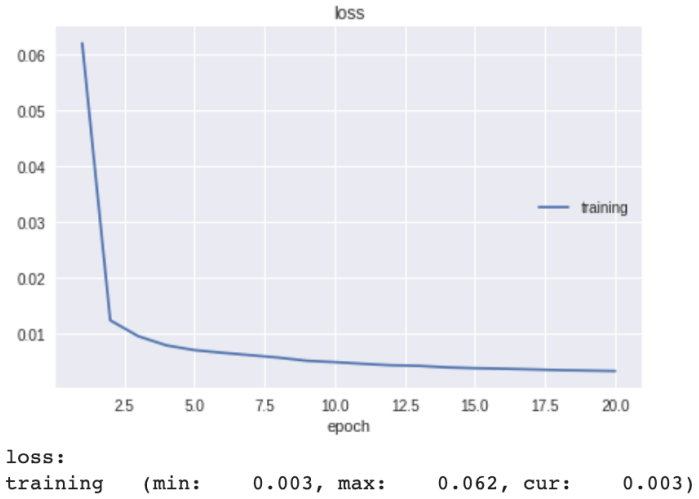


Figure 5: Training loss for the generative model on 20 epochs.

To conclude, clearly the horse shape is recognisable on the bottom images and the black wings on its side as well. At this resolution, we considered this result relatively satisfactory.

4 Improvements and obstacles

- To manually increase the training dataset we applied some data augmentation to the training dataset and concatenated the resulting sets to the original one. At first the training data included 50.000 images which we extended to 200.000 by applying: random change in brightness, contrast and saturation; random rotation; and horizontal flipping to the entire training dataset. Moreover for the Pegasus part we concatenated the test dataset as well to the training one.

- In both tasks we used a higher epoch number than originally provided: 50 for the classifier and 20 for the GAN, which provided better results but costed more time.

- To improve accuracy, every convolutional layer in our implementation was followed by a batch normalisation and a ReLU layer.

- Most of the obstacles occurred during the implementation of the classifier, because even though we intended to follow a state-of-the-art model, we had to derive our own parameters, number of layers and method choice, which was highly time consuming and the results were poor in most cases. Thus there is still a lot of space and need for fine tuning and improvements, which we did not have time to further continue.

- The reason for omitting the Sigmoid layer in the generative model was because it produced a lot worse results than without it. In addition, we experimented with omitting the fully connected layers from the autoencoder and concluded that having them reduced the noise

but introduced more blur to the generated image. However these all could be due to our limited understanding and experience in building a good model. Further investigation and fine tuning is recommended. See below the outcome with Sigmoid and without fully connected layers:

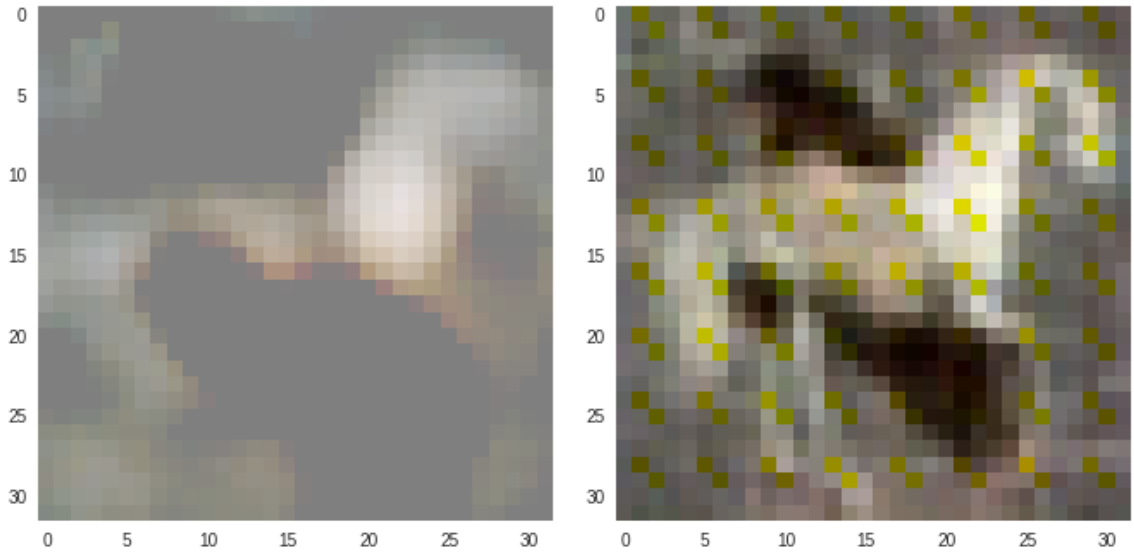


Figure 6: On the left our model with Sigmoid layer at the end of both the encoder and decoder. On the right our model without the fully connected layers.

5 Conclusion

Overall the study was successful, but the classifier could use a lot more investigation and improvement to go over 49% and perhaps using less epochs. Possibly an alternative, smarter implementation method would show better results, however due to time and knowledge constraints this was not possible.

The pegasus image is considered relatively good compared to the resolution and quality of the training data. Skipping the fully connected layers shows better edges and less blur but more noise. This could be overcome by more investigation on it.