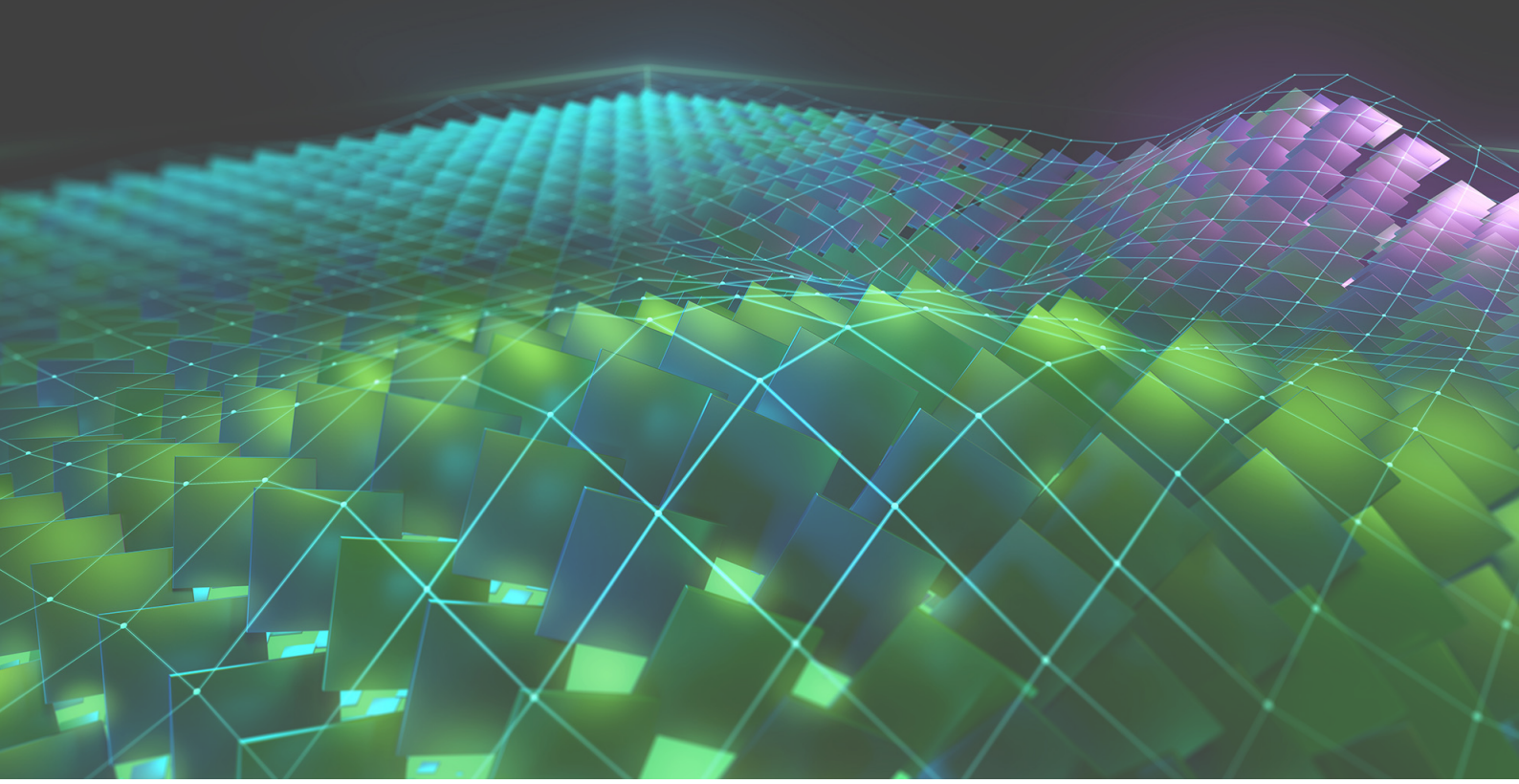


Classifying CIFAR-10 with Transfer Learning and GPU Acceleration

A DD2360 Project

Roderick Karlemstrand

KTH Royal Institute of Technology

4 February, 2021

Abstract

CIFAR-10 is a famous dataset for multiclassification problems. In this project, a Variational Autoencoder is used to classify an unlabeled sub-dataset of CIFAR-10 of 10 000 elements. Next, a small labelled dataset is used to train the classification model. The model is lastly evaluated by a test dataset. The training process takes approximately 30 minutes per epoch with CPU, so that the whole training may take days to finish one experiment. To make the research possible, a GPU accelerated environment was implemented. The results show a speedup by nearly 20 times, and the classification accuracy is around 37%.

Keywords: Image classification, Variational Autoencoder, Deep learning, Transfer Learning, GPU acceleration

Contents

Chapter 1

Introduction

Computer vision is a field of study in Machine Learning, to give computer programs the ability to see. As for humans, the light reaches the retina and signals are then sent to the brain for further processing. Human brains can connect sight with other senses, like touch, sound or smell. This makes it easy to extract certain things from the sight, to identify them. For example, a child may recognise a bike after seeing a few of them. However, it is not easy for computer programs. The pictures are just like random numbers to them, and it would take thousands of training examples for a Machine Learning model to be able to identify one certain item.

1.1 Problem Statement and Motivation

For supervised learning, the dataset has to be labelled and this is often done by humans. It is a very tedious and time-consuming job. Consider also another scenario that for autonomous driving cars, the computer program should be able to identify humans, objects, vehicles and many other things in the environment. And the input data is streaming. It would not be feasible to set labels on every one of these for every single frame. It may be possible to solve the problem in a more efficient way. In this project, a new learning algorithm of Transfer Learning is proposed. A VAE is used to separate the unlabeled training dataset into 10 clusters. And a smalled labelled training dataset is used to help the model associate the clustered learnt with the correct labels. There are no misclassifications (i.e. noises in the dataset). In this way, the model may achieve the

same prediction accuracy with the much smaller labelled training dataset. The research questions are:

- Is there a more efficient way, VAE, in particular, of teaching computer programs to classify images without large labelled dataset?
- How much faster does the training process run with a GPU, specifically Nvidia Tesla P100 16GB?

1.2 Related Work

Modelling is widely used in scientific researches. One way of doing it in the field of machine learning is discriminative modelling, which means it will try to predict something give certain observations. For example, giving an image of a cat or a dog, a machine learning model will predict the likelihood of the image being of a cat, or a dog; giving an X-ray image of a tumour, a model will predict if it is benign or malignant. Another way of modelling is called generative modelling. The machine learning model will try to generate some output given certain observations. Generative models may be superior over discriminative models, especially in the problem described above. In this scenario, one can use the generative model of the data to improve classification[2].

1.3 Objectives

The objectives of this project are to build a VAE and transfer learning algorithm and to accelerate the training process with GPU.

Chapter 2

Methods

The design of the VAE and neural network is presented in this chapter. It also includes the dataset and training process.

2.1 Variational Autoencoders

An autoencoder is a type of neural network that tries to extract features from the input data and stores it in *codes*. Codes are also known as abstract patterns. This technique forces the neural network to build a system to compress the data. Next, the code is fed into a decoder which tries to re-create the input data. This is called reconstruction [1]. The figure 2.1 shows a special type of autoencoders, Variational Autoencoder, whose standard variation and mean value is fixed.

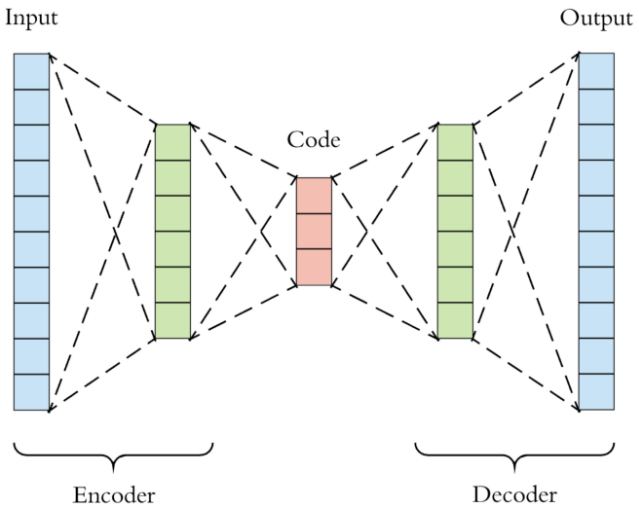


Figure 2.1: An exampel of a Variational Autoencoder

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 32, 15, 15]             896
              ReLU-2          [-1, 32, 15, 15]               0
            Conv2d-3            [-1, 64, 7, 7]          18,496
              ReLU-4            [-1, 64, 7, 7]               0
            Conv2d-5           [-1, 128, 3, 3]          73,856
              ReLU-6           [-1, 128, 3, 3]               0
            Conv2d-7           [-1, 256, 1, 1]         295,168
              ReLU-8           [-1, 256, 1, 1]               0
           Flatten-9                 [-1, 256]               0
           Linear-10                 [-1, 256]          65,792
           Linear-11                 [-1, 256]          65,792
           Linear-12                 [-1, 256]          65,792
         UnFlatten-13           [-1, 256, 1, 1]               0
  ConvTranspose2d-14           [-1, 128, 5, 5]         819,328
             ReLU-15           [-1, 128, 5, 5]               0
  ConvTranspose2d-16          [-1, 64, 13, 13]         204,864
             ReLU-17          [-1, 64, 13, 13]               0
  ConvTranspose2d-18           [-1, 3, 30, 30]           6,915
          Flatten-19                [-1, 2700]               0
             ReLU-20                [-1, 2700]               0
           Linear-21                [-1, 3072]       8,297,472
================================================================
Total params: 9,914,371
Trainable params: 9,914,371
Non-trainable params: 0
```

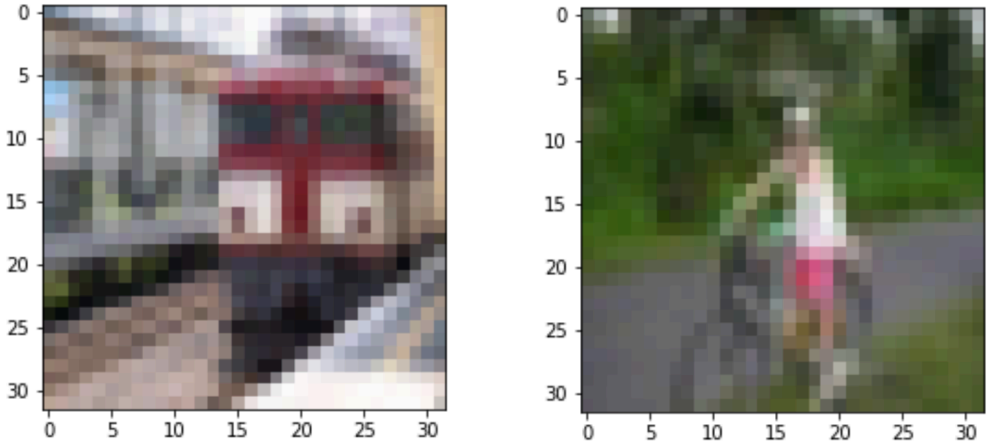
Figure 2.2: An exampel of a Variational Autoencoder

In this project, the architecture of the VAE is designed as table 2.2. The input data is images of batch size 32. Each of them has 3 colour channels and for each channel, there are 32x32 pixels. 4 convolution kernels are applied in turn to the input images, with stride size 2 and no paddings. And the activation function is ReLU. The data is then passed into the bottleneck, to force the network to find a pattern. The bottleneck is consist of Linear-10, Linear-11 and Linear-12. For the decoder, everything is reversed. As shown in layer 13 - 20. The Linear-21 will output the generated data into a simple fully connected neural network. It has 2 layers with 100 neurons each, and the output layer has 10 neurons, hence the model will predict the likelihood among 10 different classes.

2.2 Dataset

CIFAR-10 is a famous dataset for solving classification problems. It has 60 000 colour images with resolution 32x32 [3]. This dataset is suitable for researchers who have computers with limited computing power because of the low resolution. Typically in the

industry, it would require a computer with dual Xeon processors and 10 graphics cards to do experiments in different design of machine learning models without taking years. In this project, a similar data set is provided by an external sponsor, who prefer rather not to be disclosed. It also has 10 classes 2.3, by the number of labelled images is only 1 500. There are also 10 000 unlabeled images for unsupervised learning.



(a) A image of a train (b) A image of a bicyle

Figure 2.3: Two examples from the dataset

2.3 GPU Acceleration

The training of the VAE initially took place on a laptop with an Intel Core i5-7300HQ CPU, and it took about 30 minutes per epoch. Training 200 epochs would take several days. Then the training was tested on another laptop with i7-8750H, which has 6 core, 12 thread. But the temperature went so high and the fans started spinning at high RPMs. So it wasn't an optimal solution. And this is where GPU comes in handy in convolutional neural networks. The training was done using an Nvidia Tesla P100 with 16GB of graphics memory. Small adjustments were made to move the model and input data forward to the GPU.

Chapter 3

Results and Discussion

The training of the VAE and classifier went within 30 minutes and successfully. The validation loss went downwards very quickly and settles down after 50 epochs, to as low as 1.9. The prediction accuracy on the test set was up to 37%, giving randomised 1024 images where the classes are equally distributed. The model is better than random guess and considering the labelled data is only 1 500 images, the aims are reached and in the 10-classification problem, VAE does help to bring up the prediction accuracy.

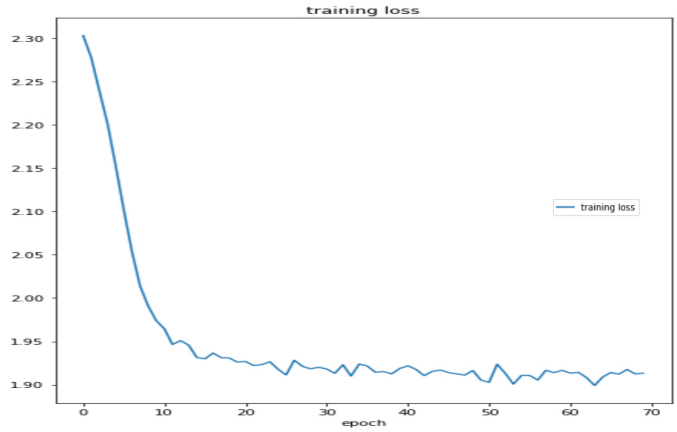


Figure 3.1: The Mean Square Error of the classifier on validation set

A comparison between CPU and GPU was also made. The CPU environment was provided by the Center for High-Performance Computing at KTH Royal Institute of Technology. The supercomputer features 128GB of memory and two Intel Xeon E5-2623 v3 3500 MHz processors, each of them has 4 cores and 8 threads. The python program for training the VAE could utilise up to 8 threads. Each epoch has 196 iterations and the

training speed was around 1 iteration per second. The whole training process is therefore calculated to be 11 hours 3.2. Unfortunately, the program caused high CPU usage and was killed by a server administrator before the training was finished. The GPU accelerated environment features 16GB of memory and two virtual cores of a Xeon E5 processor at 2.20 GHz. It has one Nvidia Tesla P100 16GB GPU installed. It only required 9 seconds per epoch and training took less than half an hour 3.3.

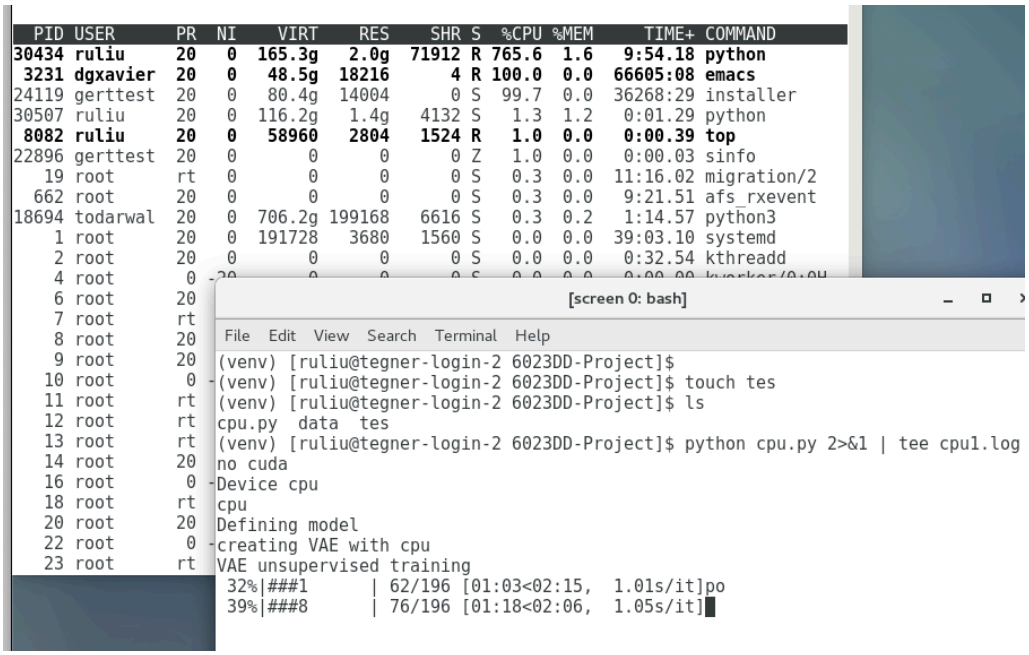


Figure 3.2: CPU consumption and time per iteration with CPU

```
VAE_tmp4 - VAE_Loss/train - 38921.297443000636 - Epoch: 82
                                                  Train Epoch: 83 Loss: 38881.11526426977
VAE_tmp4 - VAE_Loss/train - 38881.11526426977 - Epoch: 83
                                                  Train Epoch: 84 Loss: 38861.758769132655
VAE_tmp4 - VAE_Loss/train - 38861.758769132655 - Epoch: 84
                                                  Train Epoch: 85 Loss: 38897.96240732621
VAE_tmp4 - VAE_Loss/train - 38897.96240732621 - Epoch: 85
                                                  Train Epoch: 86 Loss: 38880.0939194037
VAE_tmp4 - VAE_Loss/train - 38880.0939194037 - Epoch: 86
                                                  Train Epoch: 87 Loss: 38894.61006257972
VAE_tmp4 - VAE_Loss/train - 38894.61006257972 - Epoch: 87
                                                  Train Epoch: 88 Loss: 38857.923559072064
VAE_tmp4 - VAE_Loss/train - 38857.923559072064 - Epoch: 88
                                                  Train Epoch: 89 Loss: 38804.32787089445
VAE_tmp4 - VAE_Loss/train - 38804.32787089445 - Epoch: 89
100%|██████████| 196/196 [00:09<00:00, 22.95it/s]
```

Figure 3.3: Time per iteration with GPU

As the classification neural network does not have something remarkable, it may be a good direction in future work. For example, adding dropout layers, experimenting different combinations of neurons in each hidden layer, the number of hidden layers, the activation function in each hidden layer may result in much better prediction accuracy on the test

data set. It is also a common practice to add some regularisations into the hidden layers. It can be applied to the kernel, the bias or the output. There are also various regularisations, such as weight decay. By fine-tuning the parameters, it would make it possible to reach the full potential of the VAE architecture.

There is also a limitation from the dataset. As we can see in figure 1, the picture is very blurry and the picture is about a human and a bicycle. This may confuse the machine learning model. Simple models are not resistant to noises or mislabelling in the training set, hence the performance would suffer from that.

References

[1] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[2] Diederik P Kingma et al. "Semi-supervised learning with deep generative models". In: *arXiv preprint arXiv:1406.5298* (2014).

[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: `http://www.cs.toronto.edu/~kriz/cifar.html`.