Deep Learning for Visual Computing

Assignment 1

Dario Giovannini
Julia Putz

April 2023

Contents

1 Introduction 1

2 Image Classification 1

3 Loss functions and Cross Entropy 1

4 The train-test-validate paradigm 2

5 Linear model results 2

1 Introduction

In this exercise, classification of images from the CIFAR-10 dataset was performed using a linear classifier, implemented using pytorch. Only images showing cats and dogs were used, reducing the number of classes to two.

2 Image Classification

Image Classification is the task of assigning a label to an image, usually from a limited set of labels called classes. This is usually done to identify the content or subject of the image, and trained using supervised methods, such as a labelled training set. The granularity of the classes can vary between datasets, for example the distinction between different breeds of dogs might be made, or they might all be labelled as "dog". When predicting class labels, they often come with a confidence value attached, and a prediction is sometimes correct if the true label is in the top 2 or 5 most "likely" classes, for example. Especially for datasets with many classes, some of which can be quite similar to each other, this may be necessary to more accurately evaluate model performance.

3 Loss functions and Cross Entropy

A parametric model, such as the one trained for this assignment, requires a loss function to measure the model performance on the training data, which can then be used to optimize the parameters and thus achieve iterative improvements as more data is shown and evaluated.

Cross-Entropy is one such loss function. It measures how different the predicted class probabilities are from the true class label. It uses a concept from information theory called Entropy, which describes the information gain from sampling a random process. It is defined as

$$H(\mathrm{u}) := -\sum_{t=1}^{T} u_t \log_b(u_t),$$

where u_t are the components of the mass function of a probability distribution, and b the logarithm employed, which is arbitrary for the purpose of Cross-Entropy but classically $b = 2$ is used in information theory. T is the dimension of u.

In Cross-Entropy, two probability mass functions u and v are compared:

$$H(\mathrm{u}, \mathrm{v}) = -\sum_{t=1}^{T} u_t \log(v_t)$$

Thus, the more different u and v are, the higher the Cross-Entropy H - it measures the dissimilarity.

In order to apply Cross-Entropy to image classification, it is necessary that the class labels are one-hot encoded (so are represented as a vector of length T, where T is the number of classes present in the classification problem, with 0 everywhere but the true label, which is 1). The class-scores similarly need to be returned by the model as a vector of length T, with each class being assigned a confidence score. A perfect prediction where the true label is assigned a confidence of 1 and all others a confidence of 0 would result in a minimal Cross-Entropy, while any deviation raises the Cross-Entropy and thus results in a higher loss.

4 The train-test-validate paradigm

The train-test-validate paradigm splits the total data available for training into three parts, with varying ratios according to need and size of the dataset. The training set is used for tuning the model parameters by comparing predicted values to the ground truth, applying the loss function and optimizer and thus iteratively improving the model. The validation set is used to evaluate a trained model by comparing the predicted to the true values for the entire set, and calculating an appropriate metric such as accuracy for classification tasks. The test set is used as a sort of meta-validation set, and is important when tuning hyperparameters or repeatedly training a model in epochs to evaluate the ultimately selected model. It is essential that neither the validation data and especially not the test data are seen during the training process - the concept only works if unseen data is used for evaluation, as the purpose is to train models that perform well on new data.

5 Linear model results

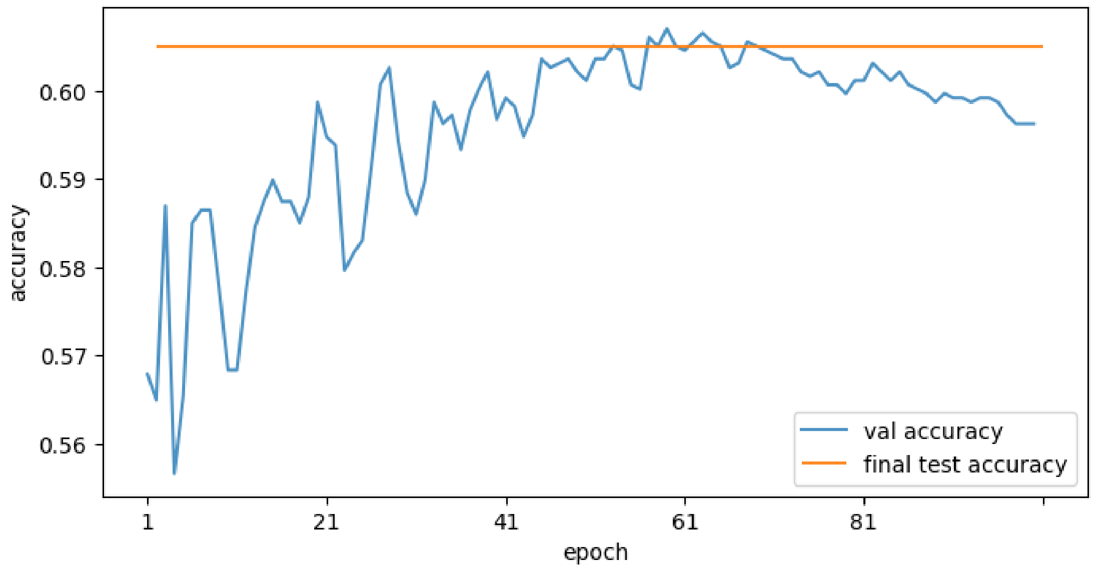

Figure 1: Validation accuracies across epochs for the linear model

The linear model trained for this assignment is simple by design, using the LinearClassifier implementation from pytorch with a single linear network layer with 32*32*3 (the image size) inputs and 2 outputs (the number of classes). It achieves a final test accuracy of 0.605, compared to a best validation accuracy of 0.607 in epoch 59 - see also figure 1. The final test accuracy is good compared to the average validation accuracy, but slightly below the best validation accuracy.

These accuracies are modest, with ∼60% only being slightly better than random guessing for a binary classification task. The model complexity, being a linear model, is very limited and thus one cannot expect particularly great results for a difficult task like image classification. The model limitations are also illustrated by the progression of validation accuracies through the epochs - the model reached a peak accuracy around the 60th epoch, and declined again from there. This suggests the model reached a saturation point where it could not learn more from the data.

Each epoch was trained using the full training dataset, as batch size was set to the maximum. The loss function used is Cross-Entropy as discussed, and the optimizer is Adam. All optimizer parameters were left at their default values. Some tuning of hyperparameters and experimenting with other options for the optimizer might bring slight improvements to the results, but a linear model is naturally limited and it seems unlikely that it would achieve very good results with any amount of fine-tuning.