Assignment 1

183.663 Deep Learning for Visual Computing (2022S)

Group 9
Yu Kinoshita (01623806), Michael Köpf (01451815)

1 Introduction/Notes on the Implementation

The objective of this assignment was to

- implement a dataset loader for a subset of the CIFAR-10 dataset (cats and dog images)
- implement a (mini) batch generator
- implement utility functionality (functions for image pre-processing and tracking the accuracy during training/validation)
- train and evaluate a linear classifier on the CIFAR-10 subset using `PyTorch`

The *dataset loader* is parameterizable and may either load the images of the training, the validation, or the test set.

The *(mini) batch generator* takes a dataset loader as an input and splits its images into batches of a given size. Additionally, the generator includes the functionality of applying a chain of image pre-processing steps and shuffling, which affects the order in which images are returned. In our implementation, the dataset is (re-)shuffled on iterator creation (i.e., when training a classifier, the order of images in two consecutive epochs should be – in general – different if shuffling is enabled).

For the training procedure of the linear classifier, we chose Adaptive Moment Estimation (Adam) as optimizer. The model was trained on the training set; the validation set was used to estimate the generalization error. Finally, the model with the highest validation accuracy was evaluated on the independent test set. We also submitted the model in the ZIP file (`model.pt`).

2 Q&A

Why do general machine learning algorithms (those expecting vector input) perform poorly on images? What is a feature, and what is the purpose of feature extraction? Explain the terms low-level feature and high-level feature.

The main reason why general machine learning algorithms perform poorly on images is the feature extraction step. A feature encodes information on the content of an image or a part thereof. Feature extraction is the task of obtaining features from images. The purpose is to extract (hopefully) expressive, and desirably also low-dimensional, features.

Images can be viewed as tensors, for example, a gray-scale image is a 2-dimensional tensor; an RGB image is a 3-dimensional tensor. Since a general machine learning model expects a vector as input, the images need to pre-processed somehow. The most naïve approach is to simply 'flatten' the image tensor, which results in poor performance because we look at single pixel values (and not features) and, additionally, the model is not able to capture the spatial relationship between the pixels (due to the flattening). When using feature extraction without employing Deep Learning, we have to get along with 'low-level feature extractors'. Such feature extractors detect minor local features in an image (e.g., edges). However, this features are often not discriminative enough. Another drawback is that a single image can have hundreds (or even

thousands) of low-level features, which need to be combined in a way that they result in a single vector – again, the spatial relationship may be lost.

As outlined above, low-level features are features that only encode minor local low-level characteristics of an image, for example, lines or edges. The extracted features are semantic-agnostic/task-agnostic, this is, the features do not encode any semantic information (e.g., presence of tiers in an image of a car). In contrast, high-level features are task-specific. The methodology that enables high-level features is Deep Learning, where the feature extractor is trained to recognize these task-specific features. Many different deep neural network architectures exist. One architecture that revolutionized computer vision are Convolutional Neural Networks (CNNs). A CNN is a feature-extractor-classifier-combination that exploits the spatial relationship between pixel values and is capable of learning high-level features via a composition of low-level features. Typically, the front convolutional layers learn filters for finding low-level features, while the rear convolutional layers learn high(er) level features, for example, detecting object parts like body parts or faces.

What is the purpose of a loss function? What does the cross-entropy measure? Which criteria must the ground-truth labels and predicted class-scores fulfill to support the cross-entropy loss, and how is this ensured?

In the context of Machine Learning, a loss function (also called cost function) is a function that measures the performance of a model on a given input (dataset) w.r.t. to the model's parameters.

The cross-entropy between $\mathbf{u}, \mathbf{v} \in \mathbb{R}^T$ is defined as $H(\mathbf{u}, \mathbf{v}) = -\sum_{t=1}^{T} u_t \ln v_t$ and measures the dissimilarity between the expected and predicted probabilities of a single sample.

The encoded vectors of ground-truth labels and predicted class-scores have to be valid probability mass functions, this is, all vector entries have to be non-negative and the sum of the individual entries of a vector must be 1. To ensure this, the ground-truth labels are normally one-hot-encoded, which results in a vector where exactly one entry is 1 and all other entries are 0. One possibility to ensure this for the predicted class-scores is the application of the Softmax function. In case the vector of a one-hot-encoded ground-truth label and the vector of the predicted class-scores (after Softmax was applied) are equal, the cross-entropy is 0.

What is the purpose of the training, validation, and test sets and why do we need all of them?

In general, each sample (in the exercise a sample corresponds to an image) is contained in only one set, this is, the training, validation and test set are disjoint.

The training set is used to 'train' a model (i.e., fit the model's hyperparameters).

The validation set is used to (continuously) estimate the generalization error of a model w.r.t. to its current hyperparameters during or after training and thus is used for hyperparameter optimization or model selection. It is important to note that the samples in the separate validation set may not be used as training data. Alternatively to a separate validation set, also other approaches exist, for example, (k-fold) cross validation.

The independent test set is used once for the final performance evaluation of a model after the training/hyperparameter optimization is completed. In particular, the test set is needed since we want to know how well our model performs on completely 'unseen' data. Thus, the test should provide an unbiased estimate of the final model fit. This cannot be done with neither the training set, since it was used to fit the hyperparameters, nor the validation set, since it was used to optimize them.

3 Obtained Results using a Linear Classifier

We made use of the shuffling functionality of the batch generator and trained the linear classifier for 150 epochs. To ensure reproducibility, all seed values were initialized accordingly. As optimizer, we chose Adam with the following parameters:

- Learning rate $\eta = 0.001$
- $\beta_1 = 0.9$, $\beta_2 = 0.999$
- $\epsilon = 1 \cdot 10^{-8}$
- Weight decay 0

Figure 1 shows the average loss per epoch on the training set and the accuracy on the validation set over all 150 training epochs. We can see that the loss two times increases and afterwards decreases significantly in the beginning. From approximately epoch 30 on-wards, the loss steadily decreases and seems to level out. The accuracy on the validation set shows a high oscillation until epoch 40.

The *highest validation accuracy* is achieved in *epoch 42 (60.6%)*. Thus, the model of epoch 42 was used for the final evaluation of the test set. The *accuracy on the test set* is *61.2%*. Thus, the validation set seems to marginally overestimate the generalization error, however, the obtained accuracies are quite similar. Two reasons that may explain the higher accuracy on the test set:

1. The samples in the test set are 'easier' than the one in the validation set.
2. The distribution of cats and dog images in the two sets is different. The validation set contains less cat than dog images (1016 vs. 1025 images) while the number is equal in the test set (1000 images per class).

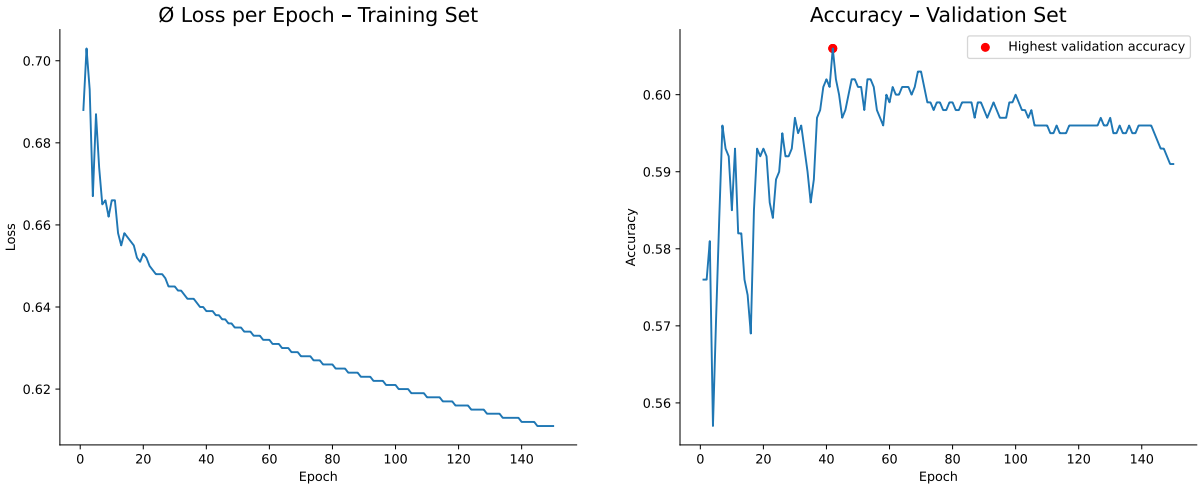

Figure 1: Average loss per epoch on the training set (left) and accuracy on the validation set (right).