AiAp 2024 Graded Miniproject 2:
Deep Learning, multiclass image classification

Your AI-Applications grade will be composed of 3 grades:

- Miniproj 1, 15% weight
- **Miniproj 2, 35% weight**
- written Exam, 50% weight

This document specifies Miniproject 2

1. Important Dates:

Start: Tuesday, March 26th 2024
1:1 Meeting Monday, April 29th / Tuesday, April 30th (online, max 20min per Team, doodle)
Deadline: Monday, May 6th 2024 midnight

2. Teamwork!

- Work in teams of two people. Larger teams are not allowed. Working alone is the exception. If you want to work alone, send me an email indicating a good reason!
- Collaborate effectively! You learn more and you are faster when you discuss your ideas or problems.
- Both team members will get the same grade. We reserve the right to interview (randomly selected) team members in order to make sure no one gets a free ride! Note that fraud is not accepted at OST and may have severe consequences.

3. Specification

In this miniproject you develop a deep convolutional network (CNN) to classify images. Your task is to **define, train and compare 3 different CNN architectures on the same data**.

1. The first CNN should be very simple and **underfit**.
2. The second ANN should be too complex and **overfit**.
3. The third network has exactly the **same** architecture and number of parameters as the second, but now you add **regularization**. This network should be optimized to solve the task reasonably well

You have to use the following technologies (no exceptions):

- Python (we recommend to create a conda environment)
- Keras/TensorFlow
- Jupyter notebook

Note: This project is not specified in all details! Some tasks require self-study (we provide a few links). The Keras/TensorFlow tutorials seen in the exercises provide a good starting point (for example https://www.tensorflow.org/tutorials/images/cnn). From there, develop the project: use the internet, discuss with peers and ask questions on the Moodle Forum.

3.1. Deliverables

Implement your project in one single **Jupyter notebook**. Use Markdown-blocks to explain and discuss the code and results in the same document. Make sure the first block in your ipynb file contains the names of the team members.
Upload the following documents on Moodle:

- Your .ipynb file

- A PDF, generated from that .ipynb file.
- Do NOT upload any other files. Do NOT upload data.

3.2. Tasks

a) **Find a dataset, suitable for multiclass image classification on internet.**
You are NOT allowed to use the MNIST or fashion-MNIST data. You are not allowed to use any of the predefined datasets which can be imported directly from tf.keras.datasets. Important: be aware that searching, analysing and pre-processing a dataset is a potentially time-consuming task! Choose something relatively simple and make sure you have annotated (labelled) data! Make sure the dataset fits the task (multi-class classification, not segmentation).
Use a search engine and look for public datasets. You are likely to find things like:
   - https://www.aicrowd.com/challenges
   - https://datagen.tech/guides/image-datasets/image-dataset-for-classification/
   - https://imerit.net/blog/top-13-machine-learning-image-classification-datasets-all-pbm/
   - https://www.kaggle.com/datasets?search=image
   - http://vision.stanford.edu/aditya86/ImageNetDogs/
   - ...

   Key characteristics of your dataset:
   - RGB images of at least 4 classes, at most 10 classes.
   - at least 500 samples *per class*, at most 20'000 samples *in total*
   - Drop data if your dataset has more classes and/or samples. When deleting data, make sure you still have enough samples per class. There's no benefit (for this project) in having a large dataset. On the contrary: using too much data slows you down.
   - If your dataset consists of high resolution images, scale them down (typical image width/heights should be in the order of 32 to 64 pixels). You lose a bit of accuracy, but using larger images slows down the network training without contributing to the learning goals of this miniproject.

b) In the notebook, cite and describe your dataset properly (data source, pre-processing)
c) Split your dataset:
   - "Lock" 30% of the data aside. This is your **test set**. Don't touch it until the very end of this miniproject.
   - The *remaining* 70% of the data are your **training and validation data**.

   Typically you split your data by calling the function *sklearn.model_selection.train_test_split* . You assign the return values to different variables. "Locking away" in our context simply means that you do not access the variables `test_images` and `test_labels` during optimization

```
train_val_imgs, test_images, train_val_labels, test_labels =
                        sklearn.model_selection.train_test_split(
                                all_images, all_labels, test_size=0.3)
```

   After this, you need to split the `train_val_...` data again into `training_data` (typically 80%) and `validation_data` (typically 20%):

```
sklearn.model_selection.train_test_split(
                        train_val_imgs, train_val_labels, test_size=0.2)
```

Exploratory data analysis

d) Analyse and visualize your data set. How many samples per class do you have? Is the dataset balanced? Plot a few images. What is the range of values of the images? If it is [0 255] you need to scale or normalize the data.

Architecture 1 (underfitting):

e) Use Keras to define and train a deep convolutional network for multi-class image classification. Keep the network very simple. It should not have more than 2 to 3 trainable layers (MaxPooling or Flatten do not count as trainable layers), and no more than ~2'000 trainable parameters. Make sure you observe (and document) **underfitting**. Use MaxPooling or increase stride to decrease the number of parameters. Avoid stride>kernel-size (do you see why?)

f) Monitoring: Plot how loss and accuracy evolve over time. For plotting, you can follow the examples of the Tensorflow tutorial. Show 4 curves: loss and accuracy for training and for validation data. If you are interested, you can use the more advanced Tensorboard (but there are no extra points for this).

g) Make sure you train the network long enough (sufficient nr of epochs, typically between 30 to 200 epochs, but start with 10 to advance faster). The training-loss should "converge".

h) Plot a confusion matrix of the validation data. Make 3 different plots for 3 different values of the 'normalize' parameter (see https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)

i) Discuss your observations.

Architecture 2 (overfitting):

j) Define and train a second model. Increase the number of layers, number of filters (=kernels), the number of neurons in the dense layer etc. Make the network too complex such that you observe overfitting. Depending on the dataset, you may need in the order of 10^5 to 10^6 parameters to see overfitting.

k) As before, plot learning curves and confusion matrix. Make sure you observe (and document) overfitting.

l) Comment / discuss your model and the results.

Architecture 2 (optimized):

m) Make a copy of architecture 2 (that is, use *exactly the same* number of layers/neurons/kernels). Then optimize/regularize it:

n) Add at least one dropout layer. Set the dropout-rate to a value between 0.1 and 0.5 https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

o) Add L2 regularization to the Conv-Layers and to the Dense-Layers by setting the parameters `kernel_regularizer=l2(lambda)`. Typical values for lambda are in the order of 10^{-6} to 10^{-3} https://machinelearningmastery.com/how-to-reduce-overfitting-in-deep-learning-with-weight-regularization/

p) Try different values for dropout-rate and lambda. Make sure you see a clear effect of regularization. Try and report the following: As a test, you can use very high values for lambda and dropout-rate; the network should underfit. For very small (zero) regularization parameters, the network should again overfit. The optimal values are somewhere between these extremes.

q) Plot learning curves and confusion matrix

r) Comment / discuss your model and the results.
Note: It is possible that the overfitted network has a better *accuracy* than the regularized model (for reasons we will discuss in class). On the other hand, the *loss* should be lower.

Quantification of the model performance

s) Calculate the classification metrics Precision, Recall, F1 for the optimized model.
Use the material discussed in AI-Foundation or read articles on the web, for example:
https://www.evidentlyai.com/classification-metrics ,
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

t) Estimate the generalization error using k-fold Cross-Validation (CV) with k>=4. Report mean and std-dev of the generalization error (both, loss and accuracy). Compare these values with the previously reported loss and accuracy. Note: for CV, you create different splits of `train_val_...` You still don't touch the test-set!

Optional:
Use CV to compare/select hyper-parameters (for example, find an optimal lambda):
https://scikit-learn.org/stable/modules/cross_validation.html

u) Finally: evaluate the model performance using the **test-set**.

Discussion

v) Add a short discussion section where you comment your observations and draw final conclusions.

4. How to submit your miniproject

There will be an assignment called "Miniproj #2, Submission" on Moodle. Both team members will have to upload the (same) files.

5. Grading

The following criteria will be used to evaluate the notebook:

- All tasks implemented (complete, correct, appropriate methods)
- Quality of the comments and discussion (concise, clear, correct terminology)
- There are no extra points for going beyond the required tasks.

6. Manage your resources!

This project is worth 35% of your final grade. Manage your resources accordingly. There's no gain in using a complicated dataset. It is not a competition! Don't put effort into small increases of accuracy.

7. Respect the deadline

If you do not hand-in a Jupyter notebook (original and PDF) by the deadline, we have to give you a grade 1.0. If you are unable to meet the deadline for important reason (illness, military service, etc.), the deadline can be discussed. Contact me as early as possible!