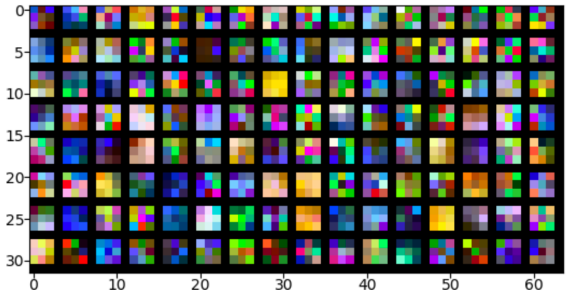
Assignment 11

Welcome to the eleventh (and last) assignment of the lecture *2D Vision and Deep Learning*. **Please read all instructions carefully!** This assignment covers some basic concepts of visualization and transfer learning. Submission is due on Monday, February 8th, 2021 at 2pm. Please note that late assignments will receive zero (0) marks, so you are strongly encouraged to start the assignment early. If you have any questions please contact Tim Heußler (`theussle@students.uni-mainz.de`).

Exercise 1 (5 points)**.** Implement a function that visualizes the *filters* of the first convolution layer of the `ConvNet` from Exercise 1.1 of Assignment 10. Your function should show all 128 filters with size 3×3 as color images (since each filter has 3 input channels). Stack these 3×3 color images into one large image. You can use the `imshow` function from the `matplotlib` library for the visualization (see the figure on the right for an example in which the filters are arranged in an 8 x 16 grid).

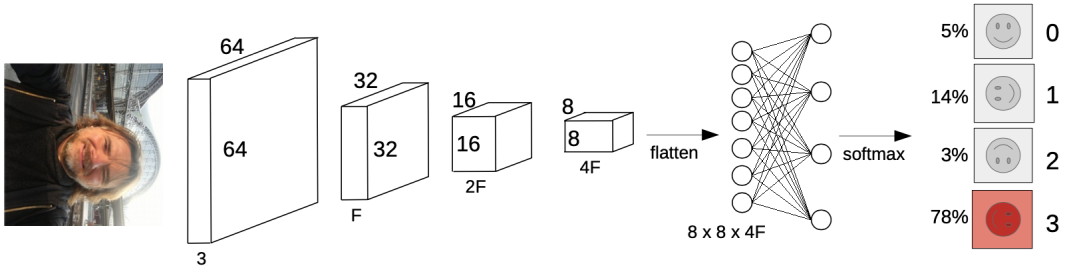


Compare the filters before and after the training. Do you see any patterns?

Exercise 2 (6 points)**.** Build and train a CNN using PyTorch that will learn to rotate photos to their correct upright orientation. Your model should basically be a 4-class classifier: Given an input image, decide wether the image is

- correctly rotated (Class 0),
- rotated left (Class 1),
- upside down (Class 2), or
- rotated right (Class 3).

Your model should look like this:



Use convolutional layers (`torch.nn.Conv2d()`) with *ReLU activation functions* and a kernel size of 3. Apply 2×2 maxpooling between the layers. Use *cross entropy loss* and *SGD optimizer* for the training.

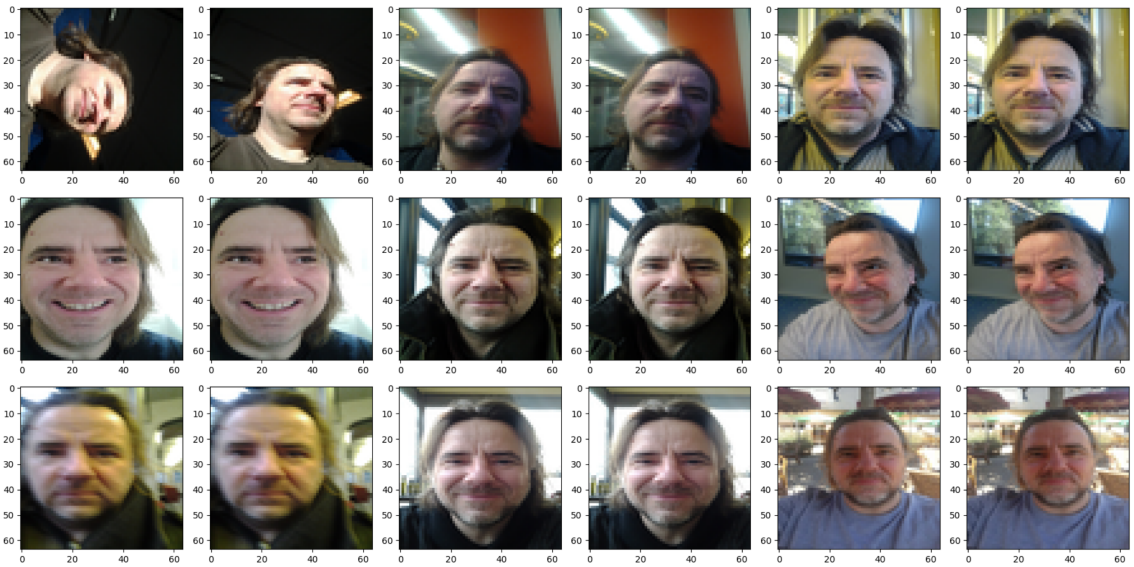
When training, track the loss and accuracy of your classifier in tensorboard, both on the training and the validation data. If you find overfitting to be a problem, you can apply a dropout layer before the fully connected layer.

You can find a simple PyTorch Script to start with as well as training and validation images in `panitzrotate.zip`.

Exercise 3 (4 points)**.** Visualize your results. Therefore

- Get a prediction from your model and rotate a few test images according to the prediction *and* according to the ground truth.
- Path both lists to the methode `plot()`, which plots the images side by side in a grid.

In the example below, the model makes on mistake (top left), the other predictions are correct.



Exercise 4 (5 points)**.** Try a second model that applies *transfer learning*. Read through the tutorial `https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html` to learn how you can load pre-trained CNNs and replace their last fully-connected layer with a new trainable one.

Use the `densenet121` model. The training code itself can remain widely the same. Fine-tune **all layers** of your network, not only the new layer you added.

Hint: You may need to extend your data transform, since most pre-trained CNNs require input data to be normalized.