Assignment #4

This assignment is due on Wednesday, November 23rd 23:59:59 via email to christian.wallraven+NNF2022@gmail.com.

Important: You need to name your file properly. If you do not adhere to this naming convention, I may not be able to properly grade you!!!

If you are done with the assignment, make one zip-file of your code directory and call this zip-file `STUDENTID1_STUDENTID2_STUDENTID3_A4.zip` (e.g.: `2016010000_2017010001_A4.zip` for a team consisting of two students or `2016010000_2017010001_2017010002_A4.zip` for a three-student team if applicable). The order of the IDs does not matter, but the correctness of the IDs does! **Please double-check that the name of the file is correct!!**

Also: Please make sure to comment all code, so that I can understand what it does. Uncommented code will reduce your points!

Finally: please read the assignment text carefully and make sure to implement EVERYTHING that is written here – if you forget to address something I wrote, this will also reduce your points! Precision is key ☺!

Part1 Pytorch CNN level 1 (40 points):

Make another notebook called `cnnA4.ipynb`.

Adapt the code from class below to use **ReLU** functions for the CIFAR 2-class problem classifying airplanes versus birds.

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 16, kernel_size=3, padding=1)
        self.conv2 = nn.Conv2d(16, 8, kernel_size=3, padding=1)
        self.fc1 = nn.Linear(8 * 8 * 8, 32)
        self.fc2 = nn.Linear(32, 2)

    def forward(self, x):
        out = F.max_pool2d(torch.tanh(self.conv1(x)), 2)
        out = F.max_pool2d(torch.tanh(self.conv2(out)), 2)
        out = out.view(-1, 8 * 8 * 8)
        out = torch.tanh(self.fc1(out))
        out = self.fc2(out)
        return out
```

Define 20 different seeds and train this network for 100 epochs with a batch size of 64, ADAM optimizer (learning rate = 1e-4), tracking training and validation accuracy in EACH epoch.

Make a nice plot of the training and validation accuracy as a function of epoch across all 20 seeds (error bars or shading!).

Answer these questions:

1. Did the network train well?
2. Do you think the variability across seeds is large or small? Why?

3. Load 100 random images from OTHER classes (so, NEITHER airplanes or birds) and put them into the network - how are the images classified? Do these results make "sense"?
4. Taking into account the results from Question 3 - would you use this network to classify birds and airplanes in the "real" world?

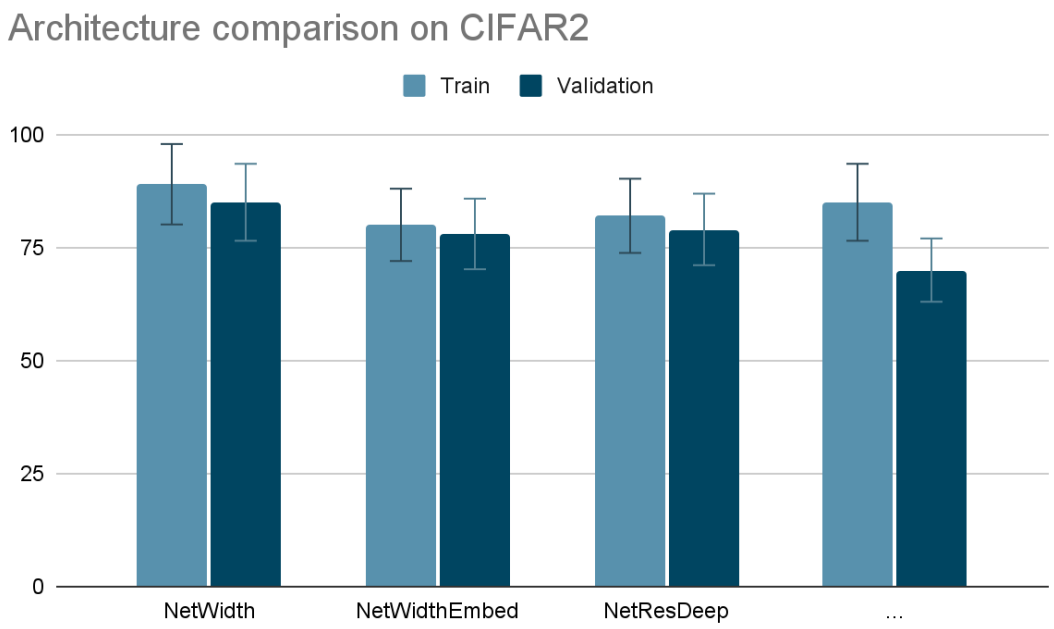
Part2 Pytorch CNN comparison (40 points):

Add all code to `cnnA4.ipynb`.

Following Part 1, now train each of the following modules with the SAME optimizer and dataloader parameters for the **same** seeds, adapting each to use the ReLU functions - use the default parameters as defined in class:

```
class NetWidth(nn.Module), class NetDropout(nn.Module), class
NetBatchNormalization(nn.Module), class NetResDeep(nn.Module), class
NetWidthProject(nn.Module), class NetWidthEmbed(nn.Module)
```

Plot the result for each of the 20 network seeds as a grouped bar graph that has on the x-axis the different networks with each network having training accuracy and validation accuracy next to each other. Remember to also include the original results from `Net` from Part 1. Please use proper error bars like so (plot is NOT complete):



Answer the following questions:

1. Are the resulting architectures (significantly) different in terms of their accuracies (for training and validation)?
2. Taking into account the number of parameters in each model, is there a relationship between training accuracy and number of parameters? (For this, you can also use another, adequate plot to support your answer)

Part3 Pytorch Multiclass (60 points):

Add your code to `cnnA4.ipynb`.

Adapt the network architecture that worked best in Part 2 to now work with the full **10 classes** from CIFAR. Keep the batch size and everything else the same and re-train the network 10 times for **200** epochs, again tracing training and validation accuracy for each epoch. Note that this will take a lot more time due to the increased size of the training set!

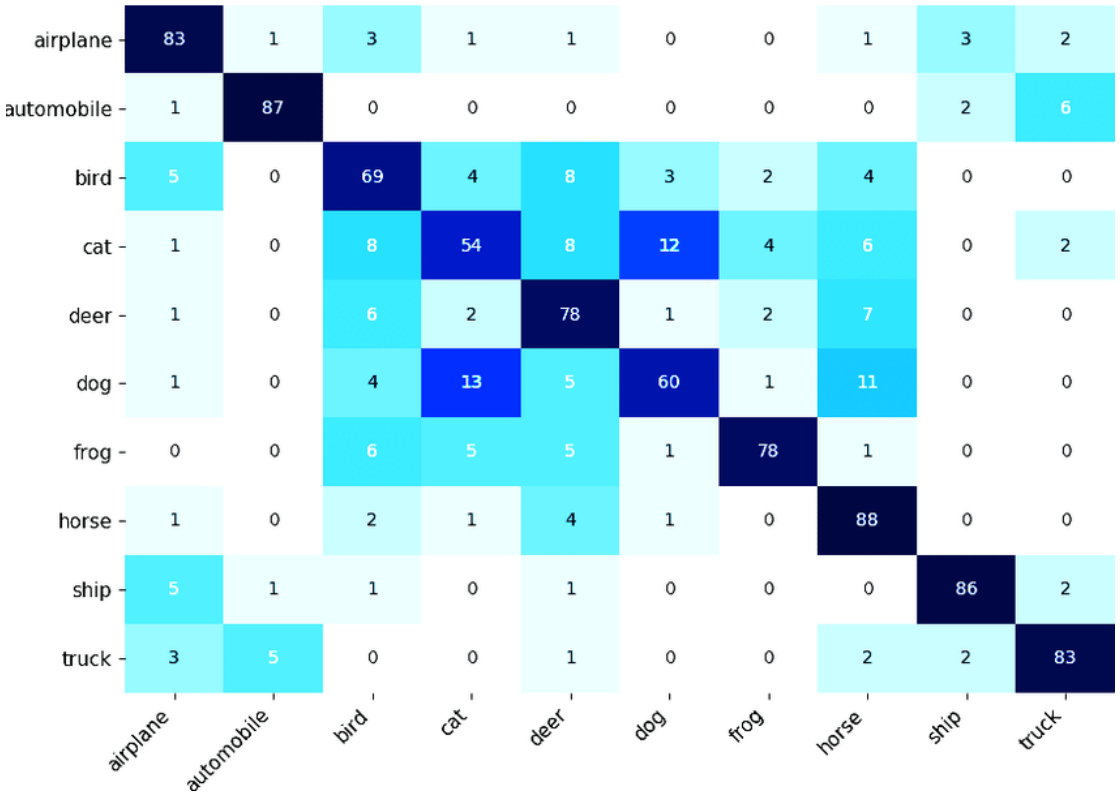
Plot the accuracy results like in Parts 1,2 as a function of epoch across seeds.

In addition, plot and nicely label a so-called confusion matrix for all 10 classes both for your final training and validation performance (so, that's two matrices).

In order to do this, you need to check which class is predicted for a given "true" label and then simply plot the result as a matrix. Make sure to use percentages!!

Note also, that you need to do this averaged across your 10 seeds!!

Something similar to this should come out, where the true class is on the bottom axis!



Now answer each of these questions:

1. Did the networks overfit?
    a. If not, how could you make them overfit?
2. Which classes can your architecture do well?
3. Which classes have the smallest / largest differences in training versus validation accuracy?
4. Do the confusion patterns in the matrices "make sense"?

Write as much as possible about each of these questions and insert your discussion into the notebook.