Instructor: Alina Vereshchaka

Assignment 1

Convolutional Neural Networks

Due date: June 13, Thu 11:59pm

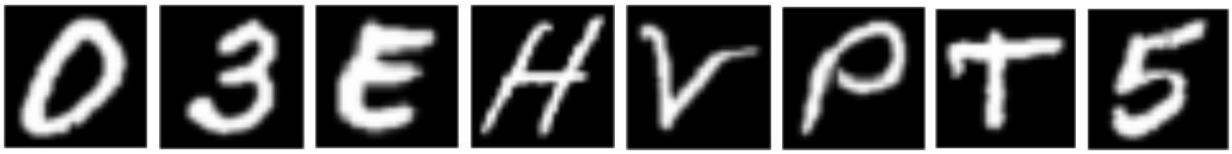
Welcome to our Assignment 1! This assignment focuses on convolutional neural networks (CNN). It consists of three parts where you practice dealing with various datasets and implement, train, and adjust neural networks.
In the first part, we will implement a basic CNN and apply optimization and in part second, we will implement the VGG-11 model.

- You can use inbuilt functions or define some functions from scratch.
- All libraries are allowed, except those with pre-trained models or predefined architectures. Submissions with pre-trained models or predefined architectures will not be considered.
- You can partially reuse related implementations that you have completed for other courses with a proper citation, e.g. “Part I is based on the CSE 574 Machine Learning Assignment 1 submission by My Full Name”

Part I: Building a CNN [40 points]

In this part we will work on a dataset containing multiple classes. You are expected to achieve a minimum accuracy of 80%.

CNN Dataset



For this part our dataset consists of 36 categories and 2800 examples for each category, thus in total 100,800 samples. Each example is a 28x28 image. A version of the dataset for this assignment can be downloaded from UBlearns (cnn_dataset.zip).

Read more about the dataset (paper)

STEPS:

1. Load, preprocess, analyze, visualize the dataset and make it ready for training. You can reuse your code from Part I.

2. Build and train a basic CNN (with max 10 hidden layers).
   Decide your CNN architecture:
   - How many input neurons are there?
   - How many output neurons are there?
   - What activation function is used for the hidden layers? Suggestion: try ReLU, ELU, Sigmoid, click here for the full list.
   - What activation function is used for the output layer?
   - What is the number of hidden layers?
   - What is the kernel size, number of filters, strides, paddings and other CNN-parameters?
   - Do you include Dropout? You can consider AlexNet to get some ideas for your network (paper).
3. Define your CNN architecture using PyTorch (basic building blocks in PyTorch) and return the summary of your model.
4. Train your model.
5. Apply one or more techniques to prevent overfitting and improve the results. Discuss the technique/s you have used and how they impact the performance:
   a. Regularization: Apply regularization techniques, such as L1 or L2 regularization, to the model's parameters.
   b. Dropout: Introduce dropout layers between the fully connected layers.
   c. Early stopping: Monitor the performance of the model on a validation set and stop training when the validation loss stops improving.
   d. Image augmentation
6.
7. Evaluate the performance of the model on the testing data. Suggested metrics:
8. Save the weights of the trained neural network. Saving and Loading Models in PyTorch
9. Visualize the results. Include the following graphs:
   a. Confusion matrices (seaborn.heatmap() or Confusion Matrix)
   b. ROC curve (receiver operating characteristic curve)
      i. Details about ROC
      ii. PyTorch ROC Function
   c. A graph that compares test, validation and training accuracy on the same plot with a clear labeling
   d. A graph that compares test, validation and training loss on the same plot with a clear labeling

Report for Part I:

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? How many entries and variables does the dataset comprise? Provide the main statistics about the entries of the dataset.
2. Provide at least 3 visualization graphs with short description for each graph.
3. Provide the model details of your CNN.
4. Discuss how the improvement tools work on CNN architectures.
5. Provide the performance metrics and analyze the results.
6. Provide graphs and analyze the results.

Part II: CNN Classification [40 points]

In this part II implement VGG. It is one of the commonly used CNN architectures. It has different versions, and, in this part, we implement the VGG-13 (Version B) and apply it to solve an image dataset containing three classes: dogs, cars and food. The expected accuracy for this part is more than 75% without optimization techniques and 80% with optimization techniques.

CNN Dataset



CNN dataset consists of 10,000 examples for each category, thus in total 30,000 samples. Each example is a 64x64 image. The dataset can be found on UBlearns.

STEPS:

1. Data preprocessing:
   a. Read, preprocess, and print the main statistics about the dataset
   b. Using any data visualization library (e.g. matplotlib, seaborn, plotly), provide at least 2 visualization graphs related to your dataset.

c. Prepare the dataset for training, e.g. normalizing, converting target to categorical values and splitting it into training, testing and validation sets.

2. Implement the VGG-13 (Version B) architecture following the proposed architecture. Be sure to use the same kernel sizes, stride, and padding as specified in the paper. You can refer to the original VGG paper for more details.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

3. Train the model on a dataset provided at UBlearns. For your dataset, adjust the size, e.g. for the input and the output layers.

4. Apply one or more techniques to prevent overfitting and improve the results. Discuss the technique/s you have used and how they impact the performance:

   a. Regularization: Apply regularization techniques, such as L1 or L2 regularization, to the model's parameters.

   b. Dropout: Introduce dropout layers between the fully connected layers.

   c. Early stopping: Monitor the performance of the model on a validation set and stop training when the validation loss stops improving.

   d. Image augmentation

6. Save the weights of the trained network that provides the best results. Saving and loading models (PyTorch)

7. Discuss the results and provide relevant graphs:

   - Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
   - Plot the training and validation accuracy over time (epochs).
   - Plot the training and validation loss over time (epochs).
   - Generate a confusion matrix using the model's predictions on the test set.
   - Calculate and report other evaluation metrics such as Precision, recall and F1 score (more details). You can either use https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html or sklearn.metrics.precision_recall_fscore_support

Notes:

- Works with in-build or pretrained models won't be considered for evaluation.
- You can partially reuse related implementations that you have completed for other courses with a proper citation, e.g., "Part I, Step 1 is based on the CSE 676 Deep Learning Assignment 0 submission by My Full Name"

In your report for Part II:

1. Describe the CNN model you have defined.
2. Describe how the techniques (regularization, dropout or early stopping) impacted the model's performance.
3. Discuss the results and provide relevant graphs:
   a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
   b. Plot the training and validation accuracy over time (epochs).
   c. Plot the training and validation loss over time (epochs).
   d. Generate a confusion matrix using the model's predictions on the test set.
   e. Report any other evaluation metrics used to analyze the model's performance on the test set.

Part III: CNN Theoretical Part [20 points]

In a convolutional neural network (CNN), suppose you're working on a computer vision project that involves classifying images of different emotions expressed by human faces. The network is designed for a multi-class classification task with 5 output classes, where each class represents a different emotion category: happiness, sadness, anger, surprise, and neutral.

CNN Architecture:

- The input consists of 32x28 RGB images. The first layer of the CNN is a convolutional layer with 10 filters, each having a size of 5x5, zero padding, and a stride of 1.
- The output values of the network represent the predicted probabilities for each emotion category and the predicted probabilities for each class are non-negative and sum up to 1.

TASKS:

1. What is the output size after the first layer?
2. How many parameters are there in the first layer?
3. What would be the output size if a padding of 1 was used instead of zero padding?
4. What would be the number of parameters if the input images were grayscale

instead of RGB?
5. Considering the above specific task and the requirement of probabilistic outputs, which activation function would be most suitable for the output layer in this scenario?
6. Prove that the activation function used here is invariant to constant shifts in the input values, meaning that adding a constant value to all the input values will not change the resulting probabilities.

In your report for Theoretical Part (Part III):

You may provide the answer using handwritten notes, typed in the MS Word or Latex. The submitted version has to be converted to a single pdf and named as avereshc_assignment1_part_III.pdf

Bonus points [max 12 points]

Implementing ResNet Architecture [7 points]

In this part implement ResNet-18 by manually defining a residual block and apply it to solve an image dataset that we use in Part I.

The expected accuracy for this part is more than 75% without optimization techniques and 80% with optimization techniques.

STEPS:

1. Implement residual blocks of ResNet, including convolutional layers, batch normalization, ReLU activation, and residual connections. You can use nn.Conv2d, nn.BatchNorm2d, nn.Sequential, nn.Identity.
2. Design a Resnet-18 model: configuration with 18-layer original ResNet paper
3. Train the model on the dataset that you used in Part I of this assignment. For your dataset, adjust the size, e.g. for the input and the output layers.
4. Apply at least two techniques to prevent overfitting and improve the results.
5. Save the weights of the trained neural network that provides the best results. Check saving and loading models (PyTorch)
6. Discuss the results and provide relevant graphs:

- Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
- Plot the training and validation accuracy over time (epochs).
- Plot the training and validation loss over time (epochs).
- Generate a confusion matrix using the model's predictions on the test set.
- Calculate and report other evaluation metrics such as Precision, recall and F1 score (more details). You can either use https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html or

sklearn.metrics.precision_recall_fscore_support

You can refer to the following ResNet Architectures.

Note: works with in-build or pretrained models won't be considered for evaluation.

ResNet Architectures

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Transfer Learning [5 points]

Use pre-trained models in PyTorch and apply them to the Food-11 dataset.

TASK:

- Work with at least three different pre-trained models, e.g. ResNet, VGG, or DenseNet. Check PyTorch documentation for more details
- Perform parameters tuning
- Evaluate the model's performance and compare the results for applying different pre-trained models

Bonus part submission:

- Create a separate zip folder named as UBIT TEAMMATE1_TEAMMATE2_assignment1_bonus.zip
  e.g., avereshc_mjadhav_ assignment1_bonus.
- Include all the files needed in the folder.
- For ResNet and Transfer Learning tasks report is not required; you can include all the comments as part of your Jupyter Notebook/Script for this task.

ASSIGNMENT STEPS

1. Submit final results (June 13)

- Fully complete all parts of the assignment and submit to UBLearns > Assignments
- In your report or Jupyter notebook include all the references that were used to complete the assignment or the task. You can include that at the end of the report.
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.
- **Add .txt file "dataset_assignment1_part1.txt" and include a link that points to the dataset you have selected. Do not submit the dataset itself.**
- Report (as a pdf file): combine the reports for all parts into one pdf file named as

  UBIT TEAMMATE1_TEAMMATE2_assignment1_report.pdf
  e.g., avereshc_mjadhav_assignment1_report.pdf
- Theoretical Part (Part IV) can be combined with a final report or submitted as a separate file named as

  UBIT TEAMMATE1_TEAMMATE2_assignment1_part_5.pdf
  e.g., avereshc_mjadhav_assignment1_ part_5.pdf
- Code (as ipynb file with saved outputs): separate file for Part I, II and Part III named as

  UBIT TEAMMATE1_TEAMMATE2_assignment1_part_1.ipynb

  UBIT TEAMMATE1_TEAMMATE2_assignment1_part_2.ipynb

  UBIT TEAMMATE1_TEAMMATE2_assignment1_part_3.ipynb
- Pickle or .h5 files with saved models that generate the best results for your model for each parts named as

  UBIT TEAMMATE1_TEAMMATE2_assignment1_part_#.h5
  e.g., avereshc_atharvap_assignment1_ part_1.h5
- For Bonus part, create a separate folder named as UBIT TEAMMATE1_TEAMMATE2_assignment1_bonus.zip
- All assignment files and bonus folder should be packed in one ZIP file named: UBIT TEAMMATE1_TEAMMATE2_assignment1.zip

  e.g. avereshc_mjadhav_assignment1.zip
- Submit your ZIP file at UBlearns > Assignments
- Verify that the submission was successful, e.g., download the submitted files and

verify they are all correct

- You can make unlimited number of submissions and only the latest will be evaluated

Notes:

- Ensure that your code follows a clear structure and contains comments for the main functions and specific attributes related to your solution. You can submit multiple files, but they all need to be labeled with a clear name.
- Recheck the submitted files, e.g., download and open them, once submitted, and verify that they open correctly

Assignments Grading

Final submission:

- Graded based on the X out of 100 points + bonus [if applicable]
- During the final evaluation, all the parts are evaluated, so please include a final version of all the parts of the assignment in your final submission.

Notes:

- Only files submitted on UBlearns are considered for evaluation.
- Files from local device/GitHub/Google colab/Google docs/other places are not considered for evaluation
- We strongly recommend submitting your work in-progress on UBlearns and always recheck the submitted files, e.g., download and open them, once submitted

Academic Integrity

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as “Copying or receiving material from any source and submitting that material as one’s own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one’s own.”. Refer to the Office of Academic Integrity for more details.

Important Information

This assignment can be done individually or in a team of up to two students. The grading rubrics are the same for a team of any size.

- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the final exam or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well the current submissions. We can see all the sources, so you don't need to worry if there is a high similarity with your Checkpoint submission.
- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit as your original work. Updating the hyperparameters or modifying the existing code is a subject to plagiarism. Your work has to be original. If you have any doubts, send a private post on piazza to confirm.
- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
    - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
    - Final grade reduction (e.g. if you obtain B, that will result in B-)
    - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

  Please refer to the Academic Integrity Policy for more details.
- The report should be delivered as a separate pdf file. You can combine report for Part I, Part II, Part III into the same pdf file. You can follow the NIPS template as a report structure. You may include comments in the Jupyter Notebook; however, you will need to duplicate the results in a separate pdf file.
- All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the information required.

Late Days Policy

You can use up to ~~5~~ 8 late days throughout the course that can be applied to any assignment-related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBlearns.

Important Dates

June 13, Thu, 11:59 pm - Final Submission is Due