**CS 4900**

**Team Project**

**750 points**

**Submission of all Components other than presentation and demo is April 30, 11:59 pm.**

Use Android Studio, python and pytorch to develop an Android application that performs automatic classification of a FashionMNIST image that is selected and loaded by the user from the storage memory of the android phone. You need to train the model yourself and are not supposed to use a pre-trained model. Please adhere to the following guidelines:

1. You need to design an app similar in functionality to the app shown in the following screenshots. For running during the in-class demo, you should use an emulator from Android Studio.
2. Create a new project, design the layout of the app using Android Studio and write relevant code in Android Studio to load and display the image selected by the user in the app.
3. Next, follow the guidelines in the next section to train a CNN model for performing image classification.
4. Write a python script to load this trained model, save it in a scripted form or as a trace and put it into the Android Studio folder for use. This python script must be commented and must be in your github repo.
5. When the user clicks the load button, the user can select and load an image into the app.
6. After loading the image user clicks the classify button to perform classification. When this button is clicked, your Android Studio code should read in the selected image so that it can be fed to the model.
7. Then, your code should load the saved model, and run the model on the image.
8. For each image, the above step gives you a probability for each class as the output. Identify the class with the highest probability and assign that class as the predicted class for the image.
9. Print the predicted class using a text in a textbox.
10. Your entire Android App directory should be in your github repo.
11. Your code should be checked in on github as well as submitted online on BlazeView. Please give me access to this **github repo**. You should check code in regularly. Please check guidelines for coding provided later.

**Training guidelines and requirements:**

1. You will perform training of the provided model. The model you need to use is: <https://gist.github.com/ahanagemini/1b3311482adfc6d00ab284601babc374>
2. Use the FashionMNIST dataset for training, validation and testing. <https://pytorch.org/vision/stable/generated/torchvision.datasets.FashionMNIST.html>
   1. You should select a subset of 20% of the images from the provided training set for validation and use the rest for training (if you are working on the options for getting a B).
   2. Make sure you sample using random sampling and there is no overlap between your validation and train sets.
   3. The classes and labels are described here:

0 T-shirt/top

1 Trouser

2 Pullover

3 Dress

4 Coat

5 Sandal

6 Shirt

7 Sneaker

8 Bag

9 Ankle boot

* 1. You should use CrossEntropyLoss.

1. The actual training process involves the following and you should have code for all these on github
   1. Write code to load the data in mini-batches (also called batches).
   2. Write code for training your model including using a non-pretrained CNN architecture for classification.
   3. Save the trained model on your local computer or Google Drive for future use
   4. Save logs for showing plots using tensorboard.

[How to use TensorBoard with PyTorch — PyTorch Tutorials 1.10.1+cu102 documentation](https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html)

1. To get a C, in the training phase, you need to show the graph for training your model with a good learning rate. You should show a loss vs. epochs plot.
2. To get a B, you need to show how to decide upon a specific learning rate for training. For this, I expect to show at least 3 graphs. The graphs can be plotted using tensorboard or otherwise but need to use the actual losses during each training epoch. The code for saving the logs or the losses must be there in your github repo. They should be loss vs. epochs plots. The three plots should be:
   1. One for high learning rate or very high learning rate training
   2. One for good learning rate training
   3. One for low learning rate training.
3. To get a B, in the training phase, you need to decide upon the number of epochs for training. Again, use one or more plots for this. You may use the training loss and try to show that you select number of epochs by training as long as the loss on the validation set keeps decreasing. The graphs can be plotted using tensorboard or otherwise but need to use the actual losses during each training epoch. The code for saving the logs or the losses must be there in your github repo.
4. To get an A, make sure you are using the learning rate, model to be used and the number of training epochs as arguments to your training script. (The grading part provides details about expectations for using arguments to the training code using either docopt or argparse)
5. If you are trying to fulfill the A requirements, you need to design two other models:
   1. A linear model that only has linear layers
   2. A CNN model that has 4 convolutional layers and one linear layer. This model needs to use an activation function other than ReLU. Add pooling layers as necessary.
6. To get an A, use your model name as an argument to the training and testing code and use the argument to select the model to be trained in your current code execution. Relevant code must be present in your github repo.

**Testing guidelines and requirements:**

1. Testing phase involves the following and relevant code for each operation should be on your github repo:
   1. Writing code to load the data for testing
   2. Load the model saved during the training phase.
   3. Write code for running each image through model to get the probability of each class.
   4. Use the output from the trained model to identify the class that has highest probability for each image.
   5. Compute per-class accuracy and mean accuracy over the whole test set.
   6. Also, compute precision, recall and F1-score, confusion matrix for each class.
   7. Compute the macro-precision, macro-recall and macro-F1-scores as well as the weighted precision, weighted recall and weighted F1-score
   8. Report all these results.
   9. If you are working for an A, you need to compare the results from the three models.

**Coding and Github Requirements:**

1. Codes for loading a dataset and for performing training and testing must be on the github repo and must be commented. All the python code should be .py files and not .ipynb Jupyter notebooks.
2. Separate python files (.py) for training, testing, and models. You will be graded on how you split into separate files.
3. A clear README.md is required with instructions for running your code. Please see the following sample: <https://gist.github.com/ahanagemini/63fdbc7f91dd2b1f9e735c0b95f6d7da>
4. All code must have proper and consistent indentation, naming style, comments.
5. You will also check in your Android Studio code. Make sure that code is also clean. You need to check in your entire Android Studio app. You will be graded on the presence and correctness of important files, the entire app code structure, MainActivity.java functionality, and mapping of class ID to classes.
6. You must put all your python code under the ‘codes’ directory on your github repo and the Android Studio Project should be a separate directory. Please note, this means there should be just two separate directories in the same repository.
7. Any team putting code under two separate repositories will be penalized. I will only be looking into one repository and one branch for each team. Besides, please note that you are creating a github repository and NOT a github project.
8. Make sure code for saving and loading CNN models are there in the repository on github.
9. Make sure the \*.py script for saving a model in a format that can be loaded in Android Studio is also there in the repo. This should contain python code, not model weights.
10. To get a B:
    1. Code must be commented and have docstrings for the file and each function.
    2. You need to divide the codes into functions.
    3. Do not write code like a script. I want to see separate functions in each file. Mostly, each function should be less than 75 lines. You will be graded on how well you divide your code into functions.
    4. Write doc-strings to specify what the function does. [How to Write Proper Docstrings for a Python Function | by Yong Cui | Feb, 2022 | Better Programming](https://betterprogramming.pub/how-to-write-proper-docstrings-for-a-python-function-7c40b8d2e153)
11. To get an A:
    1. The use of arguments for passing values to the training code is necessary.
    2. Use if \_\_name\_\_ ==” \_\_main\_\_”: to make sure the user can use the name of the file to run the code in a .py file. This is a requirement for the training and the testing .py files.
    3. You should use arguments for things like number of epochs, learning rate and other variables that can be specified by user at run-time. Please use argparse or docopt for these. [argparse — Parser for command-line options, arguments and sub-commands — Python 3.10.2 documentation](https://docs.python.org/3/library/argparse.html) or [docopt · PyPI.](https://pypi.org/project/docopt/) You should be passing the arguments as flags. Eg: python train.py --num-epochs=200 --lr=0.005
12. Code for all the training and testing requirements mentioned above must be on the github repo
13. Any code that is not checked in on github before 11:59 pm on April 27 will not be graded and will not contribute to your final grades.

**Presentation and Demo Guidelines:**

1. You must show a demo in class
2. The presentation and demo will be for 25 minutes per team on the final exam day. You may volunteer for the first day if you want to. I will give you 1% extra credit if you volunteer.
3. The demo must be shown on an emulator on your laptop:
   1. You need to figure out how to project your laptop screen on the projector
   2. Make sure you have more than one laptop on which your code app is working
   3. No excuses about laptops not working, not being able to connect to the projector, etc. will be entertained. If your demo does not work, you will get a 0 in the demo part. If your demo only loads the images, you will get partial credit.
4. Presentation (see more details in the Presentation Rubric)
   1. If you are unable to present when you are asked to do so, you will get a 0 for the presentation and the demo part unless you have written and approved documentation.
   2. You will be graded on:
      1. Structure
      2. Content (which is 75 points and includes the points that are allocated for the A and B grade requirements). This part is graded on you discussing all the work that you did outside Android Studio (which you are showing in the demo)
      3. Delivery
      4. Preparedness
      5. Enthusiasm
      6. Time limit
      7. Clarity of speech and eye contact
5. If you are absent for your presentation without written and approved documentation, you will get 0 points for the demo and presentation pert.
6. If you are absent on the day when the others are presenting, you will be penalized by 30 points unless you have written and approved documentation.
7. Each team member must speak, and time should be roughly equally shared among team members.

**Bi-Weekly Components:**

1. Week 10, 12, 14, 16 – 20 points each week based on whether you met the timeline you set, the bi-weekly submission of you and your team-mates. Week 8 will be on 15 points.
2. Each team needs to submit one BiWeekly\_Component file for each of the weeks we are doing an evaluation.
3. Submit the Bi-weekly component document - one per team. Please provide relevant details for your team's work and per-member work.  Give me details about your plan for the next 2 weeks. Follow the guidelines in the BiWeekly\_Component document online. Submissions are due by 11:59 pm on the day before the biweekly evaluation and the evaluation schedule is available online on BlazeView. There are special instructions on BlazeView for the Week 10 Assignment.

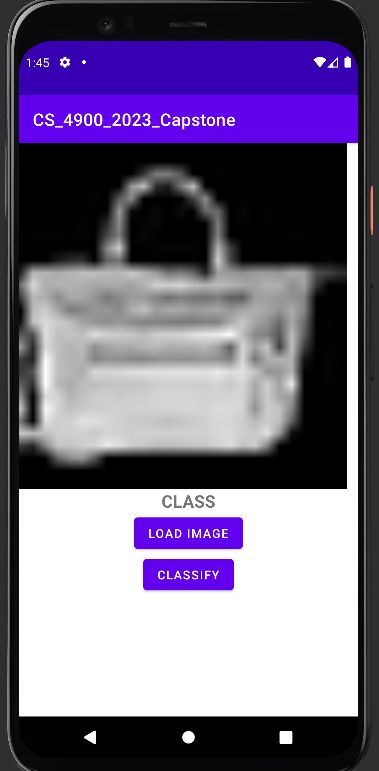
**Github and Team Contribution:**

1. Week 10, 12, 14, 16 – 20 points each week based on your github contribution, your and your team-mates’ submission and peer evaluation. Week 8 is for 15 points.
2. The Peer Evaluation should be submitted under the "Github and Team Contribution" Assignment individually by each student. Here, each student evaluates themselves and their team members. For example, if your team has 4 members, each team member should submit 4 peer evaluations. One for yourself and then one each to evaluate each of your team members. Each of your team members should also be submitting 4.
3. These are individual submissions due at 11:59 pm on the day before the Biweeky evaluation and the evaluation schedule is available on BlazeView. There are special instructions on BlazeView for the Week 10 assignment.
4. If you are absent for your Biweekly review without written and approved documentation, you will get 0 for the Github and Team contribution part for that day.

Grading Details

* Project proposal: 15 points
  + 2 – 4 pages long
  + Project Title and formatting, etc. – 3 points
  + Project summary and problem definition 3 points
  + Proposed solution 3 points
  + Timeline – 3 points
  + Reflection – 3 points
* Project proposal presentation – 10 points (10 mins per team)
  + Organization - 3
  + Clarity - 2
  + Content - 3
  + Length - 1
  + Did all team members speak? - 1
* Final Project Write-up – 100 points – Please see the details in the Project\_Documentation.docx file.
  + Presentation and Formatting – 10 points
  + Introduction – 15 points (-5 if only B requirements and -10 if only C requirements are met)
  + Method – 35 points (-10 if only B requirements, -20 if only C requirements)
  + Experiments – 35 points (-10 if only B requirements, -20 if only C requirements)
  + Conclusion – 5 points
* Final Project Demo and presentation: 125 points
  + Demo: 35 points
  + Presentation: 90 points
  + If you only fulfill B requirements, you can get at most 65 for the presentation
  + If you fulfill only C requirements, you can get at most 40 for the presentation
  + Please see the grading details provided in the grading document for this part. Besides, content includes the amount of material that was supposed to be part of the project and is covered in the presentation.
* Final Project Code and github – 300 points
  + Your code must be on github and you need to give me access to the repo. My github ID is *ahanagemini.*
  + Your code must have the following features:
    1. Code must follow all the guidelines in previous part of this document
    2. Code must be functional and work as expected.
    3. You can get at most 250 if you fulfill B requirements only and at most 200 if you fulfill C requirements only.
* Bi-Weekly Components: 100 points
  + Week 10, 12, 14, 16 – 20 points each week based on whether you met the timeline you set, the bi-weekly submission of you and your team-mates. Week 8 is 15 points.
  + Each team needs to submit one BiWeekly\_Component file for each of the weeks we are doing an evaluation.
* Github and Team Contribution: 100 points
  + Week 10, 12, 14, 16 – 20 points each week based on your github contribution, your and your team-mates’ submissions and peer evaluation. Week 8 is 15 points.
  + The Peer Evaluation should be submitted under the "Github and Team Contribution" Assignment individually by each student. Here, each student evaluates themselves and their team members. For example, if your team has 4 members, you should submit 4 peer evaluations. One for yourself and then one each to evaluate each of your team members. Each of your team members should also be submitting 4.

A screenshot of a cell phone

Description automatically generated with low confidenceA screenshot of a cell phone

Description automatically generated with medium confidence