HW1-18-794: Introduction to Deep Learning and Pattern Recognition for Computer Vision

Assignment 1: Building Neural Networks for Image Classification

Instructor: Marios Savvides

TAs: Fangyi Chen, Han Zhang, Yukai Huang, Hao Chen, Adeesh Bhargava

Due Date: October 7th, 2023

Total Points: 100

Submission: Submit your solutions, code and pdfs on Gradescope.

START HERE: Instructions

- **Collaboration policy:** All are encouraged to work together BUT you must do your own work (code and write up). If you work with someone, please include their name in your write-up and cite any code that has been discussed. If we find highly identical write-ups or code or lack of proper accreditation of collaborators, we will take action according to strict university policies. See the Academic Integrity Section detailed in the initial lecture for more information. Cases of exact same code submissions will be reported instantly.
- **Late Submission Policy:** There are a **total of 5** late days across all homework submissions. Submissions more than 5 days after the deadline will receive a 0.
- **Submitting your work:**
  - We will be using Gradescope (`https://gradescope.com/`) to submit the Problem Sets. Please use the provided template only. Submissions must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
  - **Deliverables:** Please submit all the `.py` files. Add all relevant plots and text answers in the boxes provided in this file. TO include plots you can simply modify the already provided latex code. Submit the compiled `.pdf` report as well.

NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned numbers as long as you include partial results in this pdf.

Overview

Welcome to the first Assignment! In this assignment, you'll dive into the fascinating realm of deep learning by building and training neural networks for image classification using the PyTorch framework. You'll work with two popular datasets: MNIST for image classification and CIFAR-100 for object classifcation, both of which are very popular in setting benchmarks for this task. Throughout the assignment, you'll also fine-tune your models to achieve optimal results.

Part 1: Image Classification with MNIST (POINTS: 30)

In the first part of the assignment, your goal is to construct both a fully connected neural network (part a) and a convolutional neural network - CNN (part b) for image classification using the MNIST dataset. The MNIST dataset consists of 28x28 grayscale images of handwritten digits (0-9). Your task is to build models that can accurately classify these digits. You will experiment with various architecture depths, breadths, optimizers, and learning rates to optimize the model's performance. Additionally, you will visualize the learning progress using t-SNE plots to gain insights into how the networks are learning and distinguishing between different classes.

Expected results:

- Test Accuracy of 90% for Simple Fully Connected Neural Network
- Test Accuracy of 98% for Convolutional Neural Network
- Clustering in T-SNE plots of any layer of choice
- Loss curves and understanding derived from it.
- Model summary in code submission.

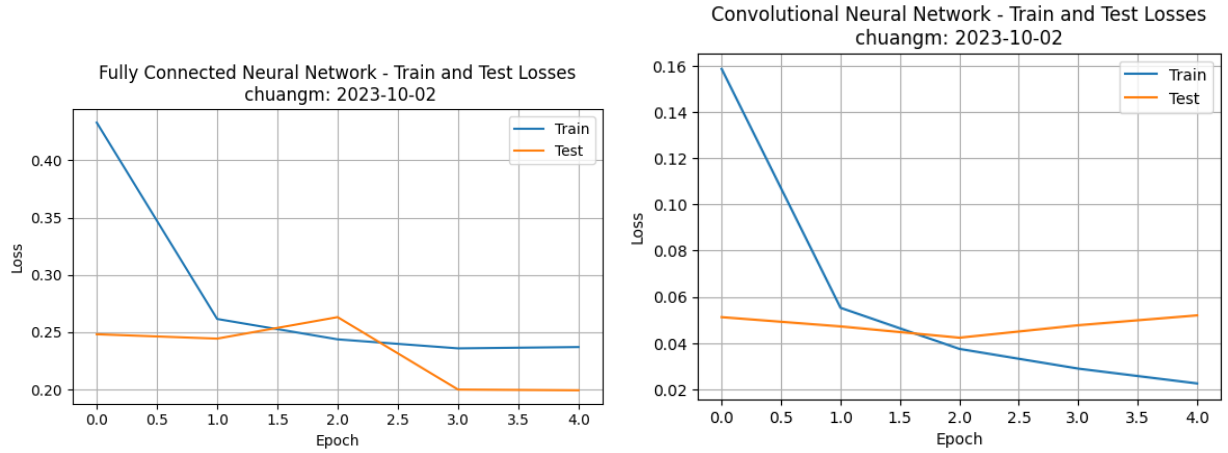
1. Train and test losses plots for Fully Connected Network and CNN Model Architectures
   FCN: Epoch 5/5 - Train Loss: 0.2370589391282722
   FCN: Epoch 5/5 - Test Loss: 0.19931943264528873

```
Epoch 1/5 - Train Loss: 0.43295426616695387
Epoch 1/5 - Test Loss: 0.24821631581475068
Epoch 2/5 - Train Loss: 0.26154867015969650
Epoch 2/5 - Test Loss: 0.24440426161738150
Epoch 3/5 - Train Loss: 0.24377605051938087
Epoch 3/5 - Test Loss: 0.26316979981715655
Epoch 4/5 - Train Loss: 0.23591282035706680
Epoch 4/5 - Test Loss: 0.19994608151791676
Epoch 5/5 - Train Loss: 0.23705893912827220
Epoch 5/5 - Test Loss: 0.19931943264528873
```

CNN: Epoch 5/5 - Train Loss: 0.022528034855788727*

CNN: Epoch 5/5 - Test Loss: 0.05195988336215419

```
Epoch 1/5 - Train Loss: 0.15850057708534365
Epoch 1/5 - Test Loss: 0.05118339798067679
Epoch 2/5 - Train Loss: 0.05522832669057587 5
Epoch 2/5 - Test Loss: 0.0471811891319512
Epoch 3/5 - Train Loss: 0.037461962062474055
Epoch 3/5 - Test Loss: 0.04229816239596727
Epoch 4/5 - Train Loss: 0.028933005297832 29
Epoch 4/5 - Test Loss: 0.04769151499018497
Epoch 5/5 - Train Loss: 0.022528034855788727
Epoch 5/5 - Test Loss: 0.05195988336215419
```



(a) Train and Test Accuracy - FCN

(b) Train and Test Accuracy - CNN

Figure 1: Train and Test Accuracy using Fully Connected & Convolutional Neural Network Architectures

2. **Understanding From Loss Curves:**
   Enter answer here: From loss we see that the training loss is similar to test loss at first, but test loss increased after 3 epochs, which means overfitting happens. The model fits well at begining but overfitting happens later. By changing loss function or adding more drop out layer can help with overfitting. Besides, can add max or mean pooling layer, which may helps.

3. Visualization of Fully Connected Neural Network Predictions
   Fully Connected Model Test Accuracy: 94.54

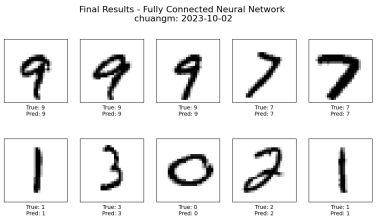


Figure 2: Fully Connected Model Architecture: Results

```
Fully Connected Model Test Accuracy: 94.54
```

Figure 3: Fully Connected Model Architecture: Results

4. Visualization of Convolutional Neural Network Predictions
   CNN Model Test Accuracy: 98.42999999999999

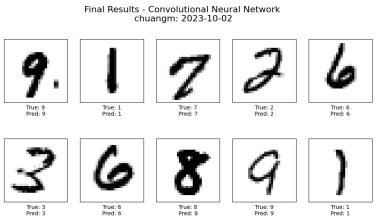


Figure 4: Convolutional Neural Network Architecture: Results

```
CNN Model Test Accuracy: 98.42999999999999
```

Figure 5: Convolutional Neural Network Architecture: Results

5. Plot T-SNE Visualisation

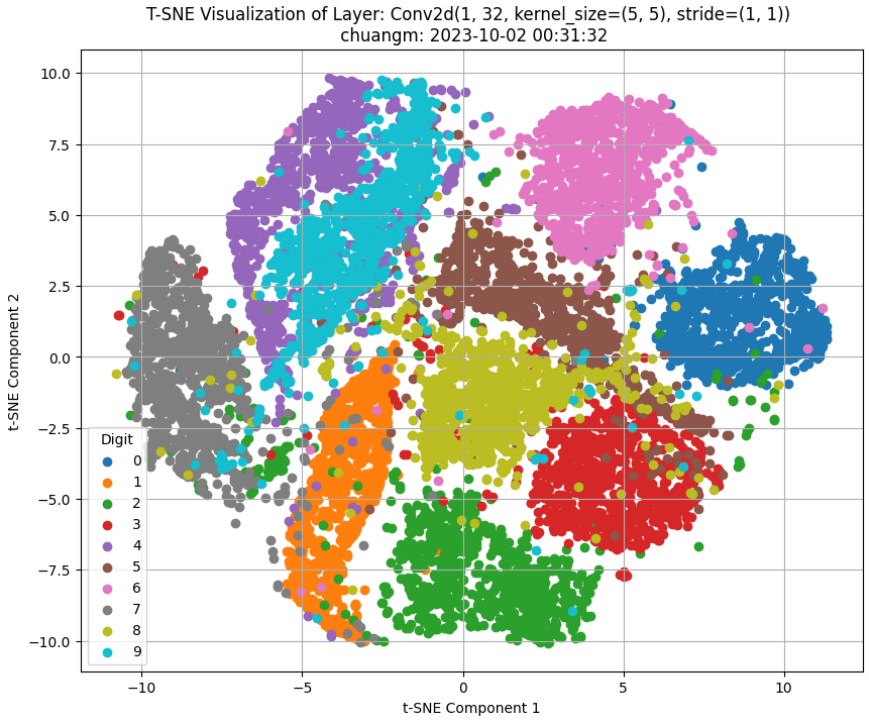


Figure 6: Convolutional Neural Network Architecture

Part 2: Object Classification with CIFAR-100 Dataset (POINTS: 30)

In the second part, your focus shifts to object classification using the CIFAR-100 dataset. This dataset contains images of various objects grouped into 100 classes. You'll again build both a fully connected neural network and a CNN, but this time, your models will classify objects from the CIFAR-100 dataset. Similar to the previous part, you will fine-tune the models, experimenting with different hyper parameters, learning rate schedulers, augmentations, initializations etc to achieve optimal results.
Note: For Deep Fully Connected Neural Network architecture, you are expected to use linear layers only.
Note: For Convolutional Neural Network architecture, you are expected to use conv and fc layers.

Expected results:

- Write code for customised loss function
- Test Accuracy of 30% for Simple Fully Connected Neural Network on CIFAR-100
- Test Accuracy of 40% for Convolutional Neural Network on CIFAR-100
- Visualisation of results (Note: There maybe mismatches at expected accuracy)
- Loss curves and understanding derived from it.
- Model Summary in code submission

1. Attach code snipped of loss function and why you chose that particular loss function:
   Note: Implement your own loss function, do not use torch.nn for this task. The goal is to understand what the loss function is doing and why is it relevant.

```
class MyLoss(nn.Module):
        def __init__(self, weight=None, size_average=None, reduce=None, reduction='mean'):
                super(MyLoss, self).__init__()

        def forward(self, predictions, targets):
                # no use torch.nn
                one_hot = torch.zeros(predictions.shape).to(device)
                tragets_one_hot = one_hot.scatter(1,targets.view(-1,1),1)
                predictions = torch.log_softmax(predictions, dim=1)
                # predictions = nn.functional.one_hot(predictions,num_classes=100)
                loss = -torch.sum(predictions * tragets_one_hot, dim=1)
                return loss.mean()
criterion = MyLoss()
```

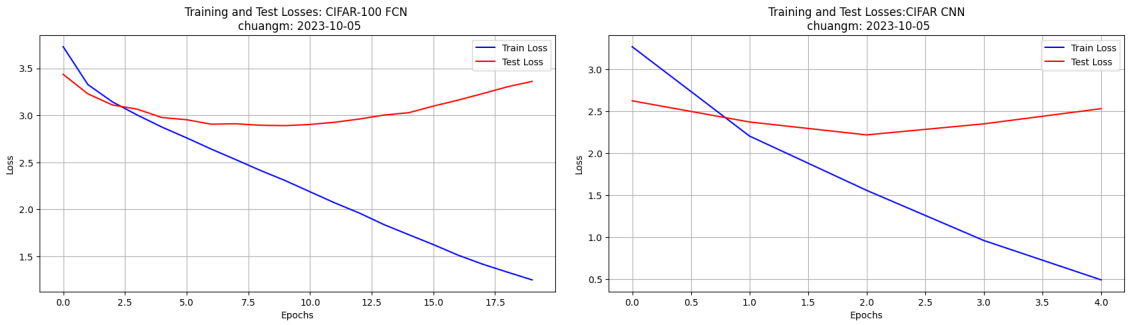
   I choose to use this function for since we are not allowed to use torch.nn. Advised by TA, I need to implement my own Cross-entropy using formula sum(p*log(q)), for it's always used and not very difficult. After reading some instruction and guidence, I realized this loss function.

2. Train and test losses plots for Fully Connected Network and Convolutional Neural Network Architectures
   FCN: Epoch 20/20 - Train Loss: 1.2510628552388048
   FCN: Epoch 20/20 - Test Loss: 3.3602068803872274

```
Epoch 1/20 - Train Loss: 3.7285638120778075, Test Loss: 3.4349446387807276
Epoch 2/20 - Train Loss: 3.326576081383259, Test Loss: 3.2283389674630136
Epoch 3/20 - Train Loss: 3.141976041562112, Test Loss: 3.109439666104165
Epoch 4/20 - Train Loss: 3.002842596119932, Test Loss: 3.06534396614998
Epoch 5/20 - Train Loss: 2.8748378220116697, Test Loss: 2.9762897612942254
Epoch 6/20 - Train Loss: 2.760315512452284, Test Loss: 2.953147423495153
Epoch 7/20 - Train Loss: 2.6410546956769645, Test Loss: 2.9057717080328875
Epoch 8/20 - Train Loss: 2.528171451042985, Test Loss: 2.910118151622213
Epoch 9/20 - Train Loss: 2.413223159435155, Test Loss: 2.893859190546024
Epoch 10/20 - Train Loss: 2.305675033260794, Test Loss: 2.890431130767628
Epoch 11/20 - Train Loss: 2.1874435251326205, Test Loss: 2.9027322113134297
Epoch 12/20 - Train Loss: 2.0698203205147667, Test Loss: 2.926344866965227
Epoch 13/20 - Train Loss: 1.9612493798555926, Test Loss: 2.9604246510062247
Epoch 14/20 - Train Loss: 1.8382295467664518, Test Loss: 3.0029853240699524
Epoch 15/20 - Train Loss: 1.7313373479087029, Test Loss: 3.0276409805200664
Epoch 16/20 - Train Loss: 1.626060790113171, Test Loss: 3.097670972726907
Epoch 17/20 - Train Loss: 1.5139514445648778, Test Loss: 3.161338071154941
Epoch 18/20 - Train Loss: 1.4173991760939284, Test Loss: 3.2307393247154867
Epoch 19/20 - Train Loss: 1.3329813323362405, Test Loss: 3.303258148727903
Epoch 20/20 - Train Loss: 1.2510628552388048, Test Loss: 3.3602068803872274
```

CNN: Epoch 5/5 - Train Loss: 0.48886833980184075
CNN: Epoch 5/5 - Test Loss: 2.52830347209979

```
Epoch 1/5 - Train Loss: 3.265734184123671, Test Loss: 2.62166770400515
Epoch 2/5 - Train Loss: 2.2027173996581446, Test Loss: 2.3698419939940143
Epoch 3/5 - Train Loss: 1.5550211030046652, Test Loss: 2.215937801986743
Epoch 4/5 - Train Loss: 0.957629092056733, Test Loss: 2.347457156059848
Epoch 5/5 - Train Loss: 0.48886833980184075, Test Loss: 2.52830347209979
```



(a) Train & Test Accuracy for FCN (b) Train & Test Accuracy for CNN

Figure 7: Train and Test Accuracy using Fully Connected & Convolutional Neural Network.

3. **Understanding From Loss Curves:**
   Enter answer here: From loss we see that the training loss is similar to test loss at first, but test loss increased after 3 epochs in CNN and after 8 epoch in FC, which means overfitting happens. The model fits well at begining but overfitting happens later. By using loss function which helps with overfitting and make NN less complex can help. Also, can add drop-out layers.

4. Visualization of Fully Connected Neural Network Predictions
   Fully Connected Model Test Accuracy: 31.72

```
Final Fully Connected Model Test Accuracy: 31.72
```

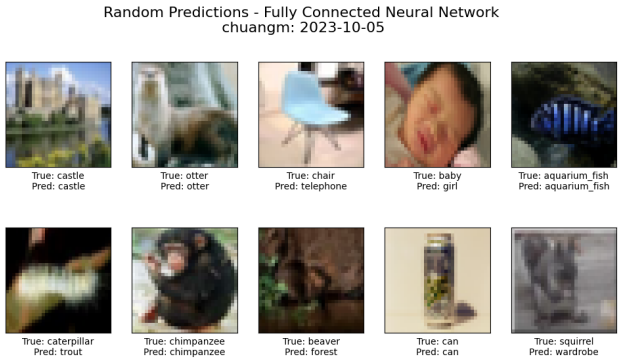


Figure 8: Convolutional Neural Network Architecture Predictions

5. Visualization of Convolutional Neural Network Predictions
   Convolutional Neural Network Test Accuracy: 44.96

```
CNN Model Test Accuracy: 44.96
```

Figure 9: Convolutional Neural Network Architecture Predictions

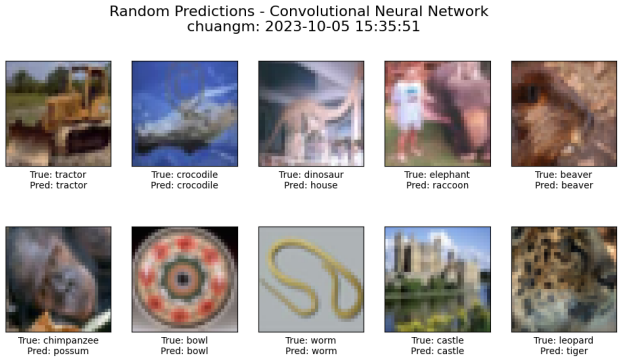


Figure 10: Convolutional Neural Network Architecture Predictions

Part 3: Fine-Tuning for Optimal Results (POINTS: 40)

In the final section of the assignment, you'll take your models to the next level by fine-tuning them and achieving benchmark performance. You'll experiment with different architectures, hyperparameters, learning rates, and optimization techniques to achieve the best possible accuracy for object classification (CIFAR-100). Your journey will involve analyzing the impact of these changes on the models' accuracy and convergence. Feel free to write your own modules, and make changes to starter code to achieve highest possible accuracy. Results are expected to be submitted in the same format as above sections however.
For your reference, here are the benchmark standards achieved by state of the art classifiers: image-classification-on-cifar-100
Note: You can implement **your own Fine tuned CNN, AlexNet, ResNET, etc.** architectures discussed in class. However, **No transfer learning or pretrained models allowed.** If, you were able to achieve 60% test accuracy with any of your previous implementations , submit same model and explain key optimizations and fine-tunings made.

Expected results:

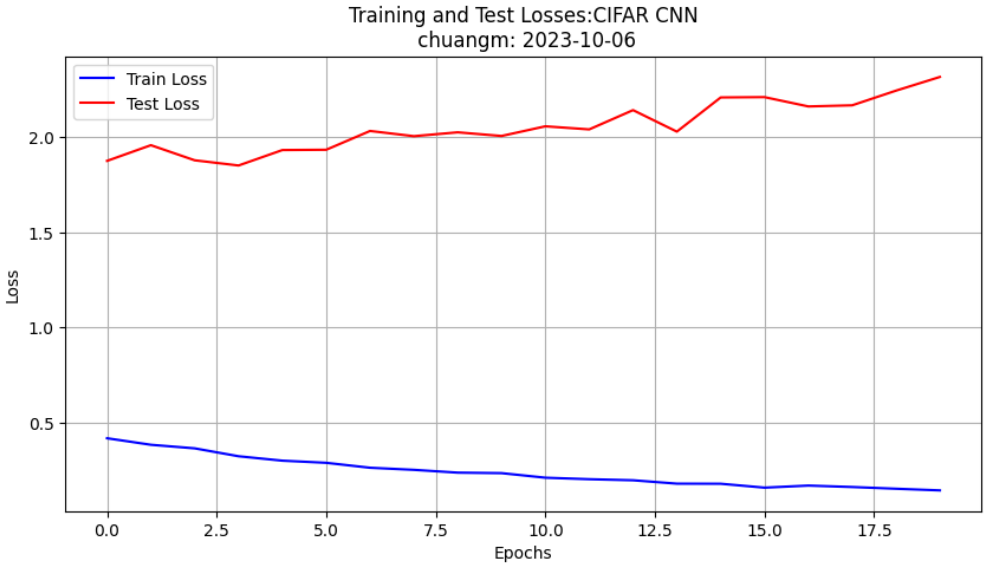
- Test and Train Accuracy of 60% or higher for full credit for your best model on CIFAR-100 dataset.
- Visualisation of results and number of correct predictions visually out of 10.
- Loss curves and understanding of model derived from it.
- Model summary in code submission

1. Train and test losses plots for the Fine tuned Architecture

   Model1:

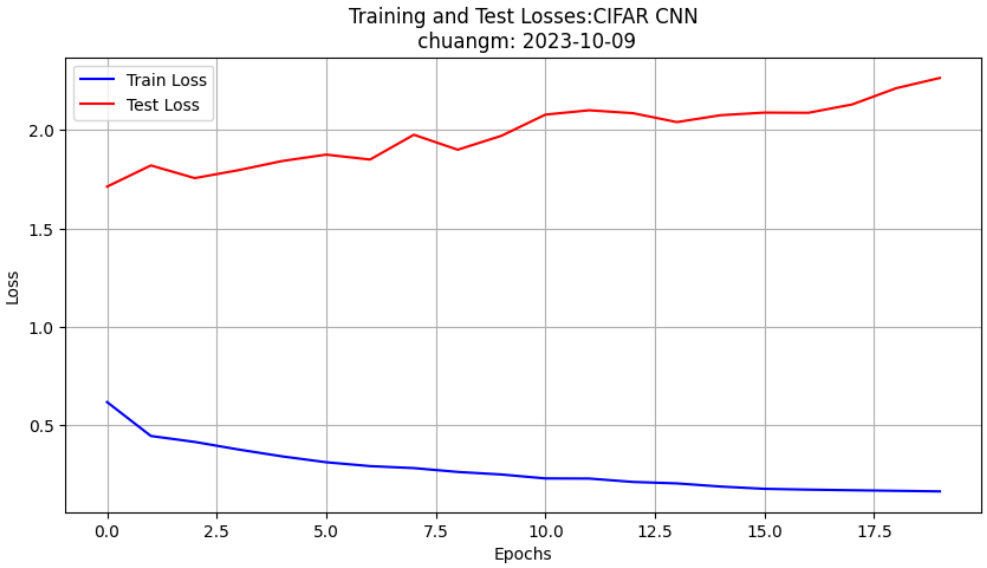
   Best Model:Epoch 20/20 - Train Loss: 0.14318444728091054
   Epoch 20/20 - Test Loss: 2.3163309007883073

```
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:152: User
  warnings.warn(EPOCH_DEPRECATION_WARNING, UserWarning)
Epoch 1/20 - Train Loss: 0.4172544264215596, Test Loss: 1.8751791715621948
Epoch 2/20 - Train Loss: 0.382961032646043, Test Loss: 1.9576490432024003
Epoch 3/20 - Train Loss: 0.36417437063492075, Test Loss: 1.8781910747289658
Epoch 4/20 - Train Loss: 0.3226342839093841, Test Loss: 1.8509423404932022
Epoch 5/20 - Train Loss: 0.29940393772356366, Test Loss: 1.9321382910013198
Epoch 6/20 - Train Loss: 0.2881734238139221, Test Loss: 1.9336567372083664
Epoch 7/20 - Train Loss: 0.26218470855026826, Test Loss: 2.032396340370178
Epoch 8/20 - Train Loss: 0.25121730482395815, Test Loss: 2.0050267308950422
Epoch 9/20 - Train Loss: 0.2364693195844183, Test Loss: 2.0252814263105394
Epoch 10/20 - Train Loss: 0.23384425416588783, Test Loss: 2.006100225448608
Epoch 11/20 - Train Loss: 0.20984349475831401, Test Loss: 2.056882306933403
Epoch 12/20 - Train Loss: 0.2020052996445067, Test Loss: 2.0406394988298415
Epoch 13/20 - Train Loss: 0.196378123395297, Test Loss: 2.1419425427913668
Epoch 14/20 - Train Loss: 0.17865276188418575, Test Loss: 2.028848487138748
Epoch 15/20 - Train Loss: 0.17816450960021846, Test Loss: 2.208622136712074
Epoch 16/20 - Train Loss: 0.15758873488069797, Test Loss: 2.2102802619338036
Epoch 17/20 - Train Loss: 0.16848658207727937, Test Loss: 2.1613036528229714
Epoch 18/20 - Train Loss: 0.16094480036776893, Test Loss: 2.1672568023204803
Epoch 19/20 - Train Loss: 0.15190907731196102, Test Loss: 2.244366708397865
Epoch 20/20 - Train Loss: 0.14318444728091054, Test Loss: 2.3163309007883073
```



Model2:

```
Epoch 1/20 - Train Loss: 0.6171858605681634, Test Loss: 1.7128946989774705
Epoch 2/20 - Train Loss: 0.4446941833106839, Test Loss: 1.8207163631916046
Epoch 3/20 - Train Loss: 0.41447800184999195, Test Loss: 1.7561959624290466
Epoch 4/20 - Train Loss: 0.3758825647587679, Test Loss: 1.796661016345024
Epoch 5/20 - Train Loss: 0.34077852941593345, Test Loss: 1.8433927118778228
Epoch 6/20 - Train Loss: 0.31109852070102884, Test Loss: 1.8756920397281647
Epoch 7/20 - Train Loss: 0.2914641181729278, Test Loss: 1.8505940943956376
Epoch 8/20 - Train Loss: 0.2813247912848482, Test Loss: 1.97678442299366
Epoch 9/20 - Train Loss: 0.2618456809040235, Test Loss: 1.9005188703536988
Epoch 10/20 - Train Loss: 0.24873925988771478, Test Loss: 1.9716935843229293
Epoch 11/20 - Train Loss: 0.22886295456971442, Test Loss: 2.0791012167930605
Epoch 12/20 - Train Loss: 0.2281324488624018, Test Loss: 2.1012216567993165
Epoch 13/20 - Train Loss: 0.2109405563345977, Test Loss: 2.086405870318413
Epoch 14/20 - Train Loss: 0.20357643489782906, Test Loss: 2.0411117702722548
Epoch 15/20 - Train Loss: 0.18749623452978476, Test Loss: 2.0761725127696993
Epoch 16/20 - Train Loss: 0.17598825677925226, Test Loss: 2.089423009753227
Epoch 17/20 - Train Loss: 0.17184809667571466, Test Loss: 2.08839613199234
Epoch 18/20 - Train Loss: 0.16873753359731364, Test Loss: 2.1309320509433745
Epoch 19/20 - Train Loss: 0.16580779052206449, Test Loss: 2.2132077753543853
Epoch 20/20 - Train Loss: 0.16289905413072936, Test Loss: 2.265640977025032
```



2. Explain all learnings and insights from fine-tunings and changes made to achieve highest possible accuracy

Enter answer here: (Quantitative results expected for atleast 3 abalation studies results)

1. I firstly uses Alex-Net and do fine-tunings based on it, but the best acc is only 52. I also sues the CNN model I built in course 11785, but the result isn't that good. After posting the question in Piazza, I knew that using Res-Net is better. So I learned how to build it while instructions from Piazza. By doing some fine-tuning, I can finally get acc at 60. This makes me know that using Res-Net can sometimes be the best choice.

2. For fine-tuning part, I found that the loss always increases with epoch increasing, so I made the channel size and the basic block size smaller. I have also tried deleting one of the layer. Though the

loss still increases later, it's better than before.

3. There are many methods we can use. By posting and get answers from Piazza, and combing knowledges I gain in course 11785, I used different scheduler, scaler and nn.init and find the ones that works well here.

4. For fine-tuning res-net, I found that the layer number, block number and channel size for each layer really matters, it can't be too small for not enough parameters. Also, it can't be too large for overfitting. I have decreased the block number and layer number, though loss looks better, the accurancy can't get 60 with 20 epochs.

5. For fine-tuning Alex-Net, I found that adding drop-out layer and make parameter around 0.2 can always make over-fitting better in FC layer.

6. I have tried many models and select two that can get acc over 60. Though the overfitting happens, the acc are better. The model with decreasing loss and smaller parameters always requires more epochs to get same result.

3. Visualization of Best Models Predictions

Model1:

Fine Tuned Convolutional Neural Network Test Accuracy: 60.62

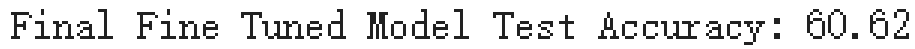


Figure 11: Fine Tuned Convolutional Neural Network Architecture: Results

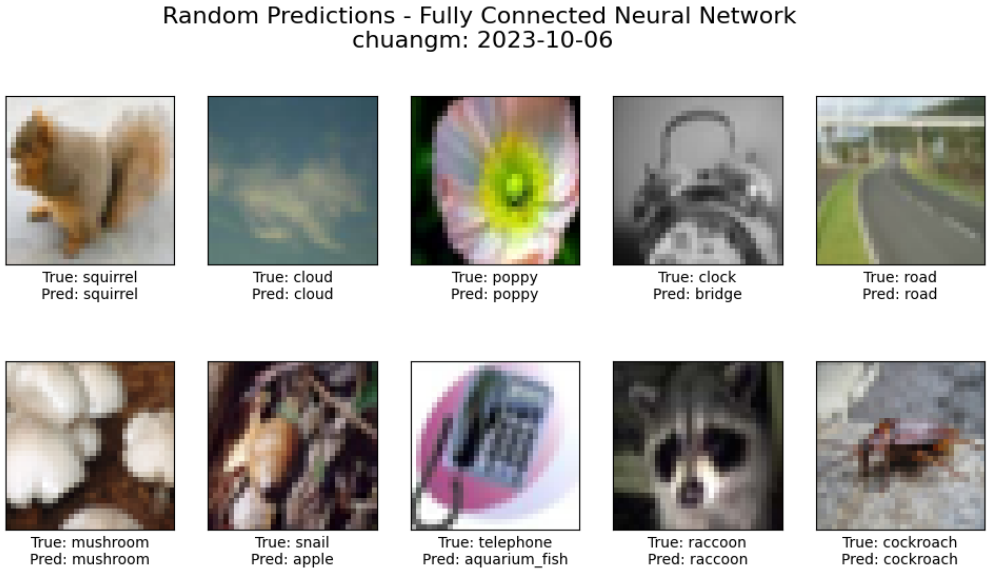


Figure 12: Fine Tuned Convolutional Neural Network Architecture: Results

Model2:

Fine Tuned Convolutional Neural Network Test Accuracy: 61.5399999999

```
Final Fine Tuned Model Test Accuracy: 61.53999999999999
```

Figure 13: Fine Tuned Convolutional Neural Network Architecture: Results

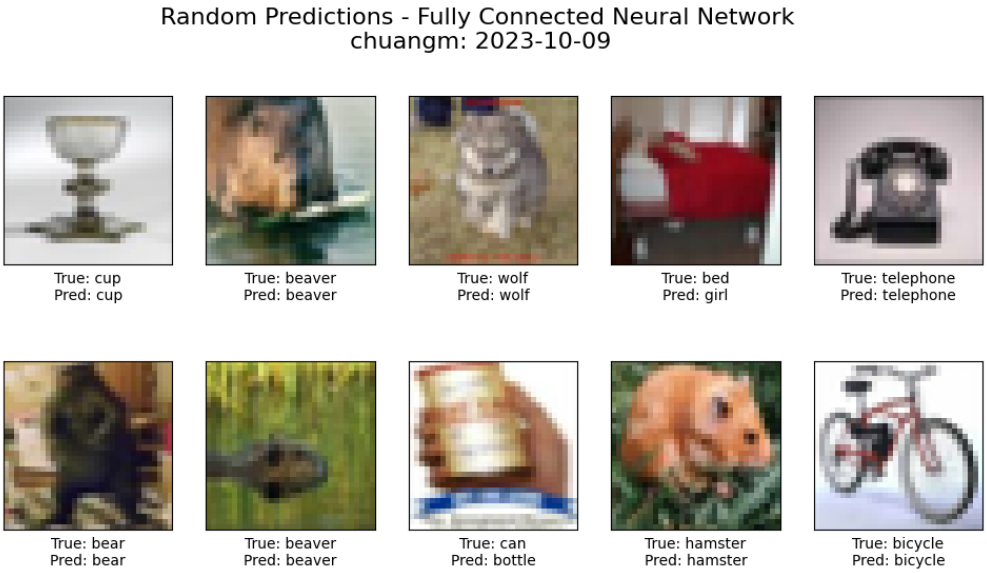


Figure 14: Fine Tuned Convolutional Neural Network Architecture: Results

Submission Guidelines

To complete this assignment, follow these steps:

- **All results in PDF should exactly match with the notebook or code submitted.**
- All plots should have their identifier (AndrewID-Date)
- Implement both the fully connected neural network and CNN for image classification (MNIST) and object classification (CIFAR-100) using PyTorch framework only.
- Fine-tune your models by experimenting with various hyperparameters and augmentations.
- Visualize the learning progress by creating t-SNE plots.
- Submit your code in either a Jupyter Notebook file or a Python script. You can submit 5 different jupyter notebooks or python scripts with the results, saving them after each run, if needed. We need to see the results in the jupyter notebook or python local folder.
- Include concise comments explaining each step and the hyperparameter choices you've made.

Important Notes

- Start early and use google colab, if you havent been able to setup your AWS credits.
- The starter code is written for CPU but can be moved onto a GPU , if available.
- Utilize only the PyTorch framework for all implementations.
- Focus on building robust models by experimenting with architectures, optimizers, data augmentations and learning rates.
- Use t-SNE plots to visually analyze the models' learning progress and class separation.

We're excited to see you explore the capabilities of neural networks in image classification. Good luck, and enjoy the assignment!

Collaboration Survey Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?

   √ **Yes**

   ◯ No

   - If you answered 'Yes', give full details, for any type of collaboration:
   - (e.g. "Jane Doe explained to me what is asked in Question 3.4")

1. I used some structures and models I built in course 11785 I'm taking this semester.
2. I get help about definition and model building by visiting OH on 10.3.
3. I get help about res-net building, useful schedular, and initialing by posting and get answers from Piazza.

2. Did you give any help whatsoever to anyone in solving this assignment?

   ◯ Yes

   √ **No**

   - If you answered 'Yes', give full details, for any type of collaboration:
   - (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. Note that copying code or writeup even from a collaborator or anywhere on the internet violates the Academic Integrity Code of Conduct.