Deep Learning Project 2: A Comparison of Architectures
Feature Extraction and Classification Performance

Fully Connected ANNs vs CNN using MNIST and fashionMNIST [composed with Keras]

IMPORTANT: **(please read and follow all steps!)**

Instructions:

1. FILES:
    a. Locate, copy, and open 'Fashion_MNIST.ipnb' in your Colab or local python workspace.
    b. The data files you'll need will be loaded as part of the script.
2. Parameter Selection:
    a. Identify the values associated with the network topology within the script.
    b. Select and modify the elements of the script required to achieve the desired performance.
3. Tables:
    a. The ***Experiments*** table (below) includes guidelines/instructions.
    b. The ***Value Table*** shows the structure of the overall project.
4. Results:
    a. **Save** a **record** of the **values** you achieved along with EACH **experimental configuration** you tested.
    b. If you choose to automate the training process, make sure you update the results for each iteration.
    c. If you choose not to save the training performance for each configuration, you should write the values to a CSV or similar for the purpose of analyzing and commenting on your work.
5. Report:

    Provide an adequately detailed report of your efforts and findings as detailed above in the "results" section.

    a. A **1** page of **discussion** – your observations when adjusting the parameters.
    b. **2-3** pages of figures, tables (not handwritten), and graphs which support the observations made in the discussion.
    c. **Colab Notebook Link** with all the tried configurations**,** which must be shared and enable public access so that your efforts are graded.
    d. Submit a **5-minute Panopto video link highlighting your report** and **include it as part of your Canvas submission**. Length is Important *(videos shorter than 5:30 or longer than 5 minutes will receive a 5 point reduction per 10 seconds)*
    e. Do not submit **GitHub Repositories, drive folders/file links, mp4 or other formats of video,** only **one document** (pdf, doc or docx) will be accepted with all the discussions, figures, tables, Colab notebook link, and the Panopto video Link.

Experiments Select or modify the values -- HOLD the other values **static** while iterating procedurally where appropriate.

Part 1 -- We're going to start out on the MNIST dataset using a state of the art NN framework.The optimizers and speed of the these methods will far outpace the basic python versions you used in the last experiment.

FC NN	
Hidden Layer Nodes	Using a procedural* search adjust the values for the number of elements in the network's hidden layers and observe the relative performance of each configuration. [This will impact the number of weights, and the complexity of the decision surface the network can approximate.] (*this can be purely **incremental** or some form of **iterative** optimization employing a meta-analysis of the architecture) ➢ Select at least **5 different values** for the number of hidden nodes per layer. (It's best if these change by some value like 10,20, or 30 for more obvious impact) ➢ (Suggested) You can choose to implement this the number of layers at the same time.
Number of Layers	The framework for the model provided in the notebook contains 2 hidden layers. Modify both the layer number and node number (in some scientific way) to determine the minimal complexity needed to achieve at least 90% accuracy. ➢ **Add no more than one hidden layer at a time** and ***iterate*** through the number of nodes for each layer procedurally – you should be able to easily produce results for **15-20** configurations (train for ~25-50 epochs)
Optimizer	select another **optimization function** and adjust the relevant parameters ➢ Try at least **5 different parameter settings** (lr,decay,reg) ➢ (Suggested) Implement some form of momentum optimization

Value Table for Your Experimental Parameters: **[EXAMPLE]**

	example	Experiment 1	Experiment 2	Experiment 3	Experiment 4
Layers	5				
Hidden Layer Sizes	64				
	32				
	24				
	16				
	10				
Training Data Division	70/20/10				
Batch Size	250				
Learning Rate	0.0002				
Optimizer	SGD				
Accuracy	50				

Use the value listed in the table above as a **guideline** for each training. Fill in your own values for all missing configurations.

Experiments Select or modify the values -- HOLD the other values **static** while iterating procedurally where appropriate.

Part 2 -- We're going to start out on the FashionMNIST dataset using a state of the art NN framework. Using a dataset FMNIST, the complexity of the data increased so does the architecture must be. Observe your performance and be ready to compare the results of your models to the same dataset using a CNN in Part 2.

FC NN	
Hidden Layer Nodes	Using a procedural* search adjust the values for the number of elements in the network's hidden layers and observe the relative performance of each configuration. [This will impact the number of weights, and the complexity of the decision surface the network can approximate.] (*this can be purely **incremental** or some form of **iterative** optimization employing a meta-analysis of the architecture) ➢ Select at least **5 different values** for the number of hidden nodes per layer. (It's best if these change by some value like 10,20, or 30 for more obvious impact) ➢ (Suggested) You can choose to implement this the number of layers at the same time.
Number of Layers	The framework for the model provided in the notebook contains 2 hidden layers. Modify both the layer number and node number (in some scientific way) to determine the minimal complexity needed to achieve at least 85% accuracy. ➢ **Add no more than one hidden layer at a time** and ***iterate*** through the number of nodes for each layer procedurally – you should be able to easily produce results for **20-30** configurations (train for ~25-50 epochs)
Optimizer	select another **optimization function** and adjust the relevant parameters ➢ Try at least **5 different parameter settings** (lr,decay,reg) ➢ (Suggested) Implement some form of momentum optimization

Experiments Select or modify the values -- HOLD the other values **static** while iterating procedurally where appropriate.

Part 3

Under the second section of the Notebook you'll see a Keras implementation of a CNN. Use that configuration as a template to start adjusting. Follow the elements below to drive for the best accuracy you can achieve!

(Industry standard is 96.7% but our tests topped out near 90% -- see if you can beat it?)

CNN with Keras	
Number and Type of Layers	The framework for the model provided is ***intentionally*** a bit suspect. Add layers (conv2d,pool,dropout. etc.) where appropriate. ➢ **one** element or layer at a time **(please try at least 10 *configurations – this does not mean 10 layer, but please employ some different orders to the layers at your disposal*)**
Kernel "Depth"	Set the values for the **number of kernels** present in the conv2d layers, pool sizes, and sequences Setting larger values will take longer to train, but may improve performance. ➢ Construct at least 5 variations on your network parameters
Batch size	Adjust the number of samples used to train the network per update. ➢ Select at least 5 values appropriate to the task and available memory
Optimizer	Using the Keras reference, select another **optimization function** and adjust the relevant parameters ➢ Try at least 2 different methods and 5 different parameter settings for each (lr,decay,reg) ➢ (optional) Implement some form of iterative hyperparameter optimization

NOTE: In an ideal world, you'll want to hold each of the following values static while changing the others and look for the place where things are performing optimally, but if you prefer to adjust the parameters by hand to try and see if you can tweak the whole system for better performance feel free to do so, but please make sure to employ some method to your madness.

Experiments Select or modify the values -- HOLD the other values **static** while iterating procedurally where appropriate.

Discussion:

1. While the MNIST dataset is perhaps the most frequently utilized dataset in ML courses, FashionMNIST is considered much more challenging. Explain why the samples in the dataset you used in this exercise seem harder to classify than the numerical identification tasks.
2. Utilizing the template attached, provide the calculations for map dimensions, number of weights and number of bias terms for your top performing CNN model.

Layer	Activation Map Dimensions	Number of Weights	Number of Biases
Input	28x28x1	0	0
CONV2D (dims)			
POOL-2			
CONV2D (dims)			
POOL-3			
FC-10			

3. Compare the results of your experiments for Part 1, Part 2 and Part 3 – use the values from your recorded model performance to generate at least (3) meaningful figures related to your results.
    a. Display the results of the **test performance** for each experiment in a single graph (**preferred**).
    b. Provide a table or plot showing how complexity of the model contributed to challenges when training both the FC and CNN implementation. (Did you overfit or stop learning?)

4. Discuss in a few sentences the results of your best and worst performing model.
    a. Were larger networks (structures with more hidden nodes) worth the trade off in training time?
    b. While the performance between FC and CNN can be large, does the training time and complexity of the CNN seem necessary for this task? Under what circumstances might this change? What if you applied augmentations? Does complexity of a dataset matter?