Quantitative Methods to Rethink Generalization

Multilayer Perceptron on Fashion-MNIST

Mao Li 3036261262

Cheng Guo 3034211964

Kexin Guo 3036385182

CS294-082: Experimental Design for Machine Learning on Multimedia Data

Prof. Gerald Friedland

Dec 17, 2021

0 ABSTRACT

In the recent decade, the massive and ever-increasing amount of multimedia data (e.g., videos, text, sounds, images) has inspired many interdisciplinary methods on data processing, analysis, recognition and inference. Machine learning is one of the most widely used approaches for multimedia-related tasks such as computer vision and speech recognition. In fact, state-of-the-art performance has been achieved by supervised or unsupervised machine learners like regression, decision trees, support vector machines, and neural networks. In this project, we applied scientific methods and experimental measurements to judge the design and limits of the machine learning models. In particular, we took an engineering approach to quantitatively measure the capacity, learnability, generalization, and resilience of a classical artificial neural network, *Multilayer Perceptron*, on a standard image classification dataset, *Fashion-MNIST*. Our work will provide valuable insights about the experimental design for neural networks on image recognition tasks.

1 PROJECT SETUP

1.1 Dataset

The dataset we used is a standard and classical image dataset called *Fashion-MNIST*. Introduced by Ha Xiao and their team, *Fashion-MNIST* comprises of 28 × 28 grayscale images of 70,000 Zalando's fashion products from 10 categories, with 7,000 images per category. The original photos of these fashion products were shot by professional photographers and then converted into unisize, grayscale images. Each grayscale image in the dataset has 28 × 28 = 784 pixels and each pixel has an 8-bit integer value ranging from 0 to 255, indicating the lightness/darkness of this pixel. In total, the training set has 60,000 images while the test set has 10,000 images. *Fashion-MNIST* is commonly used as the benchmark dataset for machine learning algorithms, especially computer vision models.

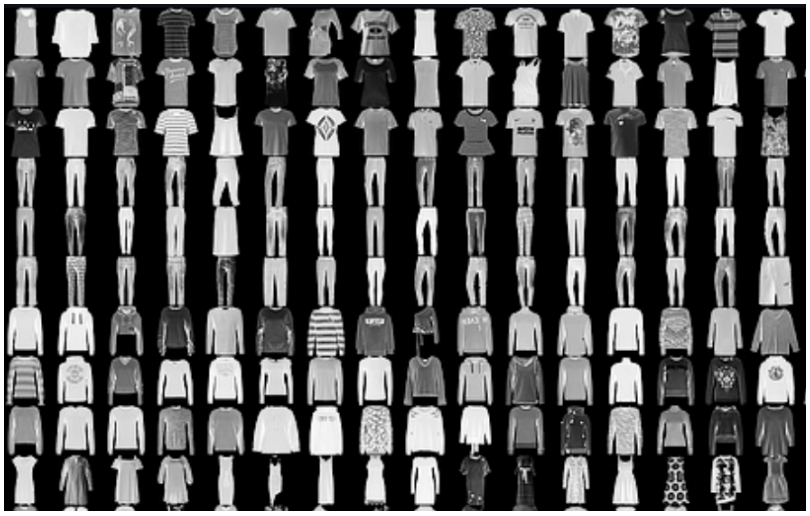

Figure 1. Samples from Fashion-MNIST dataset

1.2 Model

The machine learner we chose to measure is a classical artificial neural network: *Multilayer Perceptron* (MLP). First introduced by Frank Rosenblatt in 1962 [1] and then continuously developed by the research community of artificial intelligence, MLP is a feedforward artificial network consisting of input, hidden and output layers of neurons. Each neuron maps weighted input to output and has a non-linear activation function. Under supervised setting, the training of MLP is carried through backpropagation where weights of neurons get updated based on the error in the model's output compared to the expected result.

In this project, we used a *MLP* with 1 input layer, 2 hidden layers and 1 output layer. Its input layer takes the flattened 784 image pixel values. The fully-connected layers have 256, 128, and 128, 10 nodes respectively. The model outputs a soft-maxed probabilistic prediction of the image's class label. We chose ReLU as our activation function, cross entropy as our loss function and Adam as our optimizer.

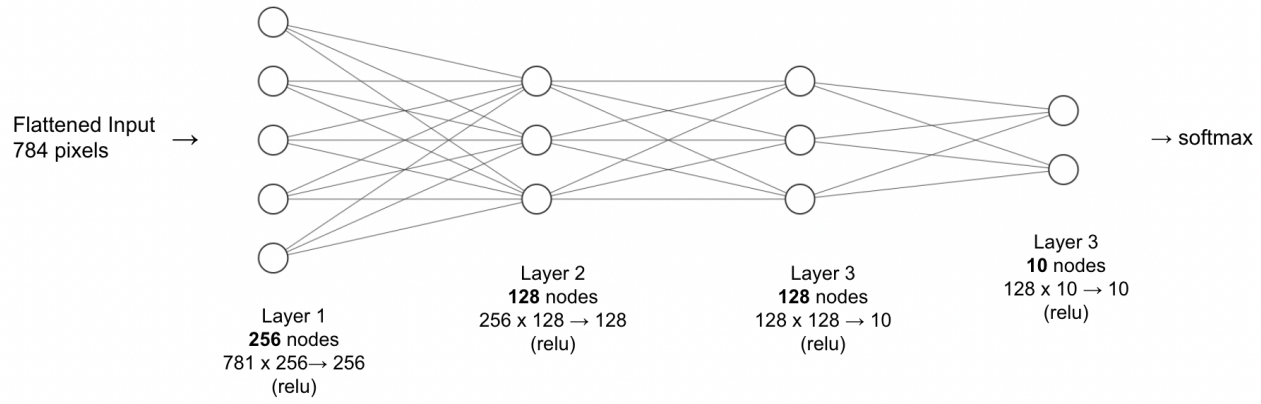

Figure 2. A four layer Multilayer Perceptron

1.3 Methods

We regard machine learning as an engineering discipline where scientific methods and experimental measurements can be utilized to evaluate the models. Our methods are primarily based on the teaching by professor Gerald Friedland in their CS294-082 *Experimental Design for Machine Learning on Multimedia Data* [2]. We used *Brainome* [3], and *TFMeter* [4] as our measurement tools. We used *Weights and Bias* [5] to run our experiments and log results.

2.DESIGN QUESTIONS

2.1 Question 1

What is the variable the machine learner is supposed to predict? How accurate is the labeling? What is the annotator agreement (measured)?

Given a sample 28 × 28 grayscale image from *Fashion-MNIST*, the machine learner is supposed to predict its class label, which is the silhouette code of the product. The labels include T-shirt/top, trouser pullover, dress, coat, sandals, shirt, sneaker, bag, and ankle boots.

We assumed the labeling to be fairly accurate, because it's manually assigned by the in-house fashion experts and then reviewed by a separate team at Zalando [6]. In addition, our team also sampled 50 images and checked their labels against our intuition. We agreed with 49 out of 50 image labels, with the exception of one shirt being labeled as dress. Thus, we argue that the labeling is 98% accurate.

The annotators labeled the images of fashion products according to their common sense and the fashion standards at the time of labeling (2017). For instance, the T-shape fabric shirts with short sleeves and a round neckline are labeled T-shirts, while the pants that cover both legs from waist to ankles are labeled as trouser pullover. Yet annotators may have some disagreements due to their different standards of fashion or common sense (e.g., some label a cloth as a dress but others think it's a shirt). In this case, the rule of majority is applied. Moreover, the annotator agreed to assign one and only one label for each image.

2.2 Question 2

What is the required accuracy metric for success? How much data do we have to train the prediction of the variable? Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do we expect bias?

We evaluate the success of our model by its classification accuracy, recall, precision, F1 score as well as the cross entropy loss [7]. The accuracy is defined as the percentage of correctly classified samples out of total samples. The recall is defined as true positive divided by the sum of true positive and false negative, $TP/(TP + FN)$. It is the proportion of actual positives that is correctly classified. The precision is defined as true positive divided by the sum of true positives and false positive, $TP/(TP + FP)$. It is the proportion of predicted positives that is true positive. F1 score is the harmonic mean of precision and recall, ranging from 0 to 1. It balances the scores of precision and recall. In addition, since our model outputs multiclass prediction probabilities, we also use categorical cross entropy loss given in [Eq 1], where M is the number of classes, y is the binary indicator on whether label c is correct classification for observation o, and p the predicted probability that observation o is of class c. The cross entropy loss measures the uncertainty of the model's prediction based on how much it varies from the actual value. These

metrics in essence all measure how accurately/balancedly our model can classify samples to each class. In our task, we assume an accuracy score of 90%+ means a successful classifier, considering that the annotation is around 98% accurate and there is noise in our images. In addition to these accuracy metrics, we should also consider the generalization score *G* given in [Eq 2]. It measures the generalization ability of our model. Ideally, a *G* score > 1 suggests that our model is learning rather than memorizing, so it can be generalized to unseen data.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \qquad Generalization = \frac{\#correctly\ classified\ instance}{MEC}$$

Eq 1. Multiclass Cross Entropy Loss Eq 2. Generalization Score

In *Fashion-MNIST*, we have 60,000 images to train our classification model. The 10 classes are balanced, with a total of 70,000 images and 7,000 images per category. The dataset only contains 1 type of media (image) and all images are grayscale, so we only have one channel of information and the modality is 1. There is no temporal information because there is no way to infer time-related features merely from the images.

We expect 732 bits of noise per image. According to prof. Gerald's paper [8], we have in general 22.4 bits of noise out of 24 bits of information in image data. Since we have 784 pixels (information bits) [9], we have 732 bits of noise per image. The noise is probably introduced in the image conversion process. The original fashion product photos were shot by professional photographers. Then they were trimmed, resized, sub-sampled and converted to grayscale. Information was lost and noise increased after the conversion.

We do expect bias in our dataset. First, there is selection / sample bias because the creator of the dataset may only select a subset of all clothing items for a single category. There may exist other clothing in the same category that are not represented in the dataset. Second, there is exclusion bias, because we have removed the color of the images. Third, there is measurement bias, because the types of cameras taking the original photos may be different.

2.3 Question 3

What is the Memory Equivalent Capacity for the data (as a dictionary). What is the expected Memory Equivalent Capacity for a neural network?

The Memory Equivalent Capacity for the data (as a dictionary) is **961,166 bits**. We applied the capacity estimation algorithm 1 presented in Chapter 9 of prof. Gerald's textbook and Piazza Post 92 [10] [17], but extended it to our balanced multiclass setting. The pseudo algorithm is shown in Figure 3, and our code implementation is open-sourced on Github [12]. In our algorithm, we assume all 784 dimensions / features / pixels of our image sample are in equilibrium and can be modelled with equal weights in the dot products. We thus ignored the training of weights by fixing them to 1 and trained only the biases. To train the bias, we create a two-column table, where each row contains the 1-weighted sums of a feature vector and its

corresponding label. Then we sort the table by the first column (1-weighted sums). Finally we iterate through the sorted table and count the need for a threshold every time a class change occurs between two neighboring rows. This is equivalent to adding a neuron with input weights 1 and giving the threshold as bias to a hidden layer of a dummy 3-layer network. We assume the machine leaner to be ideal and therefore its trained weights are maximally effective. The perfect training can cut down the number of threshold comparisons exponentially. It's like performing an efficient search algorithm. Moreover, since we are dealing with a multiclass classification problem, we computed the MEC as the weighted sum of log class thresholds multiplied by the number of input dimensions. Our algorithm gives the dataset MEC as 961,166 bits.

Algorithm 1 Calculate the Memory Equivalent Capacity needed for a **multiclass** classifier assuming weight equilibrium in a dot product.

Require: *data*: array of length n contains *d*-dimensional vectors *x*, *labels:* a column of multiple labels with length *n*.

function *memorize*((*data, labels*))
  for all *labels* **do**
    thresholds[*label*] ← 0
  end for
  for all *rows* **do**
    table[*row*] ← $(\sum x[i][d], label[row])$
    sortedtable ← *sort*(*table*, *key = column 0*)
    class ← 0
  end for
  for all *rows* **do**
    if not *sortedtable*[*row*][1] == *class* **then**
      class ← *sortedtable*[*i*][1]
      thresholds[*class*] ← *thresholds*[*class*] + 1
    end if
  end for
  mec ← $\sum_i \frac{1}{p_i} * \log_2(thresolds[class_i] + 1) * d$
end function: *mec*

Figure 3. The Capacity Requirement Estimation Algorithm for Multiclass Dataset

We then applied the four theorems in [13] to compute the Memory Equivalent Capacity for our neural network, *Multilayer Perceptron*. Our model has four fully-connected layers, with 256, 128, and 128, 10 nodes respectively. In the first layer, every node takes the flattened 784 pixels value as inputs, so each has 784 + 1 parameters. The memory capacity of the first layer is thus $784 * 256 = 200,960$ bits. In the second layer, every node takes 256 inputs so each has 256 + 1 parameters. However, as the memory capacity of neurons in a subsequent layer is limited by the output of layer it depends on and the output of first layer is 256 bits at maximum, the memory capacity of the second layer is *min(257 * 128, 256)* = 256 bits. Similar logic applies to the third and fourth layer. As a result, our model's Memory Equivalent Capacity is 200,960 + 256 + 128 + 128 + 10 = **201,482 bits**.

2.4 Question 4

What is the expected generalization in bits/bit and as a consequence the average resilience in dB? Is that resilience enough for the task? How bad can adversarial examples be? Do we expect data drift?

The expected generalization is **0.28 bits / bit**, as given by Eq. 2. We first assume the expected accuracy of a machine learner on our dataset to be above 95%, and thus the number of correctly classified instances is more than 57,000. We then use the model MEC calculated in Question 3 as the denominator. We have the expected generalization as 0.28 bits / bit. It's worth noting that the expected generalization for an ideal learner is 80.22 bits / bit, as given by *Brainome.*

The average resilience is **11.05 dB**, as given by Eq 3. Resilience measures the amount of uncertainty that can be added to an instance and it is expressed as a negative number. Since we have a positive value for resilience, each image needs to reduce on average 11.05 bits of noise (add 11.05 bits of information), in order for our model to have consistent performance. The positive resilience is certainly **not enough**, not to mention each image already has 732 bits of noise. It means that the noise in our images will impact our model's performance.

$$\text{Resilience} = 20\ \log_{10}\left(\frac{MEC}{\#correctly\ classified\ instance}\right)$$

Eq. 3 The noise resilience of a machine learner

Adversarial examples may be the noisy images that are compressed by lossy algorithms. Our positive resilience measures how bad such adversarial examples can be. We also experimented by assigning random labels to a subset of data to create adversarial examples. We observed that the model performance decreases as more adversarial examples are added to the training set.

We do expect data drift. There might be concept drift in data, meaning that we may have different fashion standards on classifying the clothes. The models trained on *Fashion-MNIST* might not have good performance on another clothing image dataset labeled by different fashion standards. There might also be covariate drift, meaning that the upstream process of creating/processing/sampling the images may change. Our model may not transfer well on datasets created by different procedures.

2.5 Question 5

Is there enough data? What does the capacity progression look like?

There **is enough** data because the capacity progression curve converges. We first generated the capacity progression on our dataset using *Brainome*. Then, as suggested by algorithm 2 in Chapter 9 in professor Gerald's textbook [10], we applied our multiclass capacity estimation algorithm [Figure 3] to compute the expected MEC for 10%, 20%, 30% … 100% of the training data. Both results are shown in Figure 5. In terms of *Brainome*'s result, we observed that as we

present 80% or more training data to an ideal machine learner, the number of decision points of this learner no longer grows. It means that 80% or less of the data is enough for the ideal machine learner to infer rules of this dataset. Any more data beyond the 80% threshold won't introduce anything new to the machine learner. In other words, a machine learner is able to identify a generalizable function from a subset of training data. Our result of expected MEC for varied proportions of training data is also consistent with that of *Brainome*. Therefore, we can conclude that our dataset is learnable.

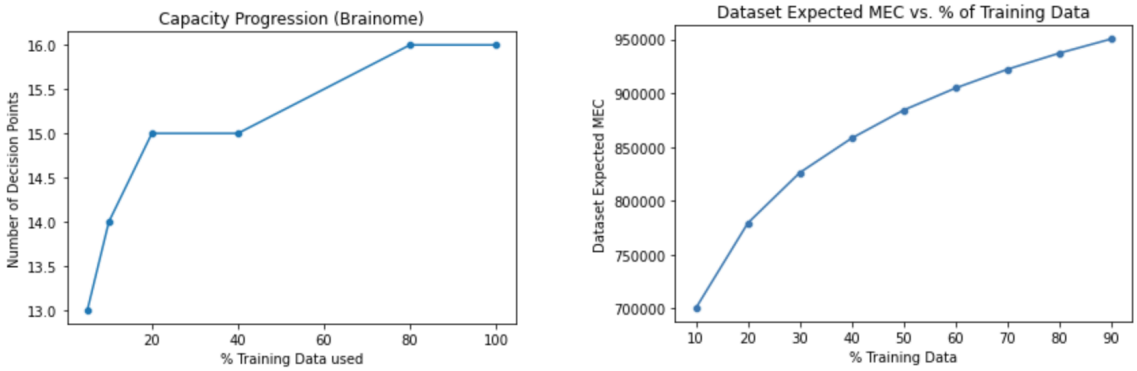

Figure 5. Capacity Progression Curves

2.6 Question 6

Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% memorization? If not, why (diagnose)?

We trained our model at its MEC (201,482 bits) on training data for 1, 10, 20, 30, 50, 80 epochs and recorded the model's accuracy on both training and independent testing sets. The result is shown in Figure 6. We can observe that after 80 epochs, the model's accuracy on the training set **reaches near 100%**, while its performance on the test no longer improves. This indicates that our model is memorizing the training data. In addition, we fixed the epoch to be 10 and then used different proportions of data to train our model. As shown in Figure 7, our model achieved a decent performance of 92% after learning on 70% of the data in only 10 epochs.

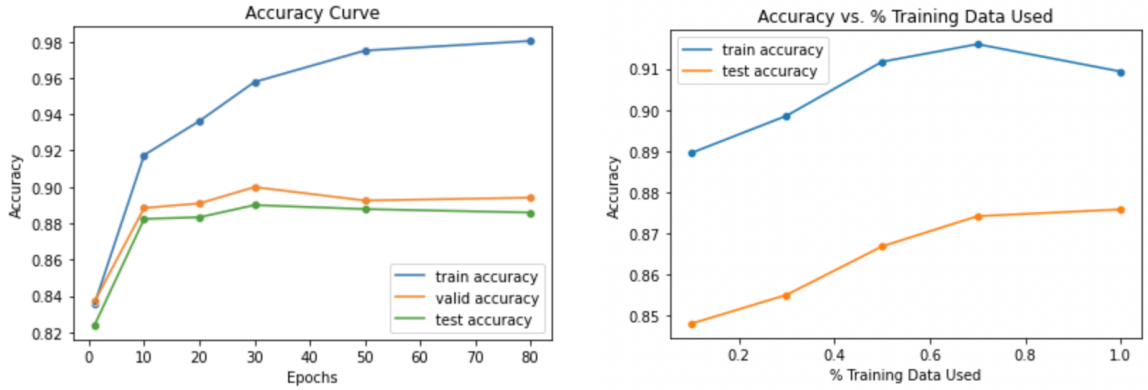

Figure 6. Model accuracy after training for epochs

Figure 7. Model accuracy vs. % training data used

2.7 Question 7

Train your machine learner for generalization: Plot the accuracy/ capacity curve. What is the expected accuracy and generalization ratio at the point you decided to stop? Do you need to try a different machine learner? How well did your generalization prediction hold on the independent test data? Explain results. How confident are you in the results?

As shown in Question 6, our current model (MEC = 201,482 bits) can successfully achieve near 100% accuracy after 80 epochs. Yet it has poor performance on test sets and its generalization constant G given by Eq 2 is very low at 0.279, meaning that this model isn't learning but memorizing. Therefore, it is a good starting point for model quantization.

We train our machine learner for generalization following these steps: 1. Remove nodes from each layer to reduce the model capacity by a factor of two; 2. Train the new model to achieve 90% accuracy; 3. Retrain iteratively with decreased capacity while recording the model's performance; 4. Stop at the minimum capacity for best test set accuracy. The models we trained with a decreasing number of parameters are listed in Figure 8. As we progressed our experiments, we recorded the model capacity and its corresponding train / test accuracy. Each training takes approximately 80 epochs to meet the standard of our performance metrics. The visualization of accuracy / capacity plot is shown in Figure 9. We observed that the model accuracy remained almost unchanged when we reduced the model capacity in the beginning. In this phase, we were improving the model's generalization without impacting its performance. However, as we continued to decrease the model capacity after some threshold, the model performance started to drastically decrease. In this phase, the model began to fail at learning. In addition, it's worth noting that our validation accuracy is very close to test accuracy. This is reasonable because both are independent unseen data from the training data. The difference is that we only used our validation data to fine-tune our hyperparameters.

	# Neurons in Four Layers	Model MEC (bits)	Train Accuracy	Validation Accuracy	Test Accuracy	Generalization
0	[256, 128, 128, 10, 10]	201482	0.982083	0.890833	0.8879	0.292458
1	[128, 64, 64, 10, 10]	100746	0.970437	0.884833	0.8819	0.577951
2	[64, 32, 32, 10, 10]	50378	0.965646	0.881667	0.8781	1.150080
3	[32, 16, 16, 10, 10]	25194	0.936583	0.878583	0.8657	2.230491
4	[16, 8, 8, 10, 10]	12602	0.900729	0.863833	0.8536	4.288506
5	[8, 4, 4, 10, 10]	6306	0.873271	0.840750	0.8381	8.308952
6	[4, 4, 2, 10, 10]	3160	0.100937	0.096250	0.1000	1.916535

Figure 8. Experiment for Generalization

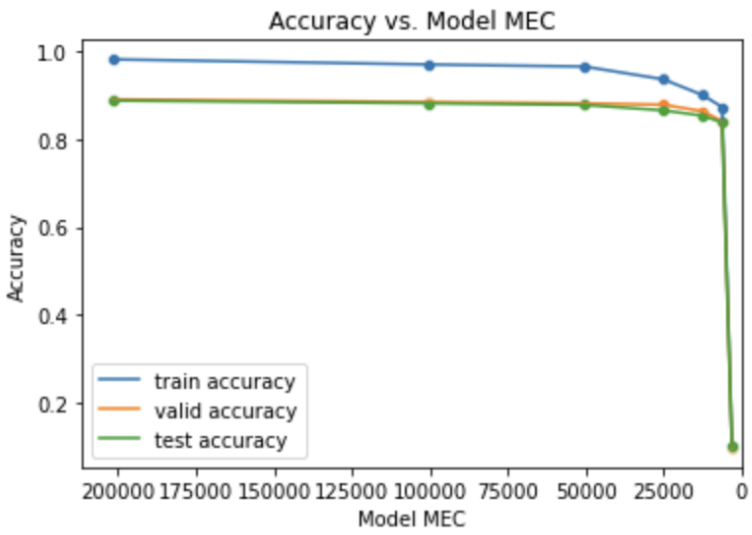

Figure 9. Accuracy vs. Capacity Curve

We decided to stop at **Model 5** whose layers have 8, 4, 4, 10 nodes respectively. As shown in Figure 9, this model is at the intersection point N_{approx} between validation and train accuracy curves. It is also the turning point where further reduction in MEC is about to drastically harm model performance. In other words, this model keeps a balanced tradeoff between accuracy and generalization. Moreover, as shown in Figure 8, the model's MEC = 6306 bits, training accuracy = 0.8732 and generalization G = 8.31 bits / bit. Having a G > 1 (or 10/9 in our multiclass case) means that this model is able to generalize and learn rather than memorizing the training dataset. Therefore, it's reasonable to conclude that our selected model achieved good performance in terms of both accuracy and generalization. Considering that our model is well-suited for this image classification task, we didn't find the need to try other machine learners.

Our generalization prediction holds fairly well on testing dataset. As shown in Figure 8 and 9, our model achieved 0.8381 accuracy on an independent testset. This result indicates that our model is able to generalize labeling functions from training data, rather than merely memorize. Although the test accuracy is lower than train accuray as expected, it is still a decent performance when compared to the more complicated models with much larger MEC. Furthermore, our optimized model MEC is only 6306 bits while the dataset MEC is 961,166 bits. This indicates that our model is able to infer general rules that represent more information in the dataset. In other words, we use less to learn more.

We are reasonably confident in our result. Our results are consistent after multiple iterations of experiments. Our results align with the intuition regarding capacity / accuracy tradeoff. Yet we admit that we haven't fully explored the effect of variables such as batch size and loss function on our experiment. Even though it's unlikely, they may lead to different results. Therefore, more quality assurance methods proposed in Question 8 should be taken in future work.

2.8 Question 8

Comment on any other quality assurance measures possible to take/ the authors should have taken. Are there application-specific ones? If time is present: How did you deal with it?

We can take quality assurance measures with respect to dataset, model architecture, training, validation and testing. First, we can use a larger dataset with more realistic images, imbalance class distributions and higher noises. Such a dataset makes it very hard for a machine learner to have good performance by random chance. Second, we can try different and more advanced model architectures such as decision trees, regression, and convolutional / recurrent neural networks. We can also compare and choose different layer structures and activation functions. By varying these features of our model, we can test if our experiment results still hold for a different setup. Third, in terms of training and validation, we can pick different batch sizes, learning rate, regularization penalty and optimizer functions. By this means, we can analyze the variables in our experiment and ensure its robustness. Lastly, we can repeat our experiments multiple times to see if results are consistent.

There are also application-specific QA measures. We can apply image quality assessment to our data in order to quantitatively measure their information and noise. We can analyze in-depth about the image conversion process of our dataset. These image-specific measures will help us better understand the learnability and generalizability of our dataset. If time is presented as a feature in our dataset, we can possibly exploit it by feature engineering it to capture the fashion trend, so that our model can detect the change of fashion standards. If it's impossible to do so after trial and error, we will simply ignore the time feature.

With these quality assurance measures, we can have a comprehensive and quantitative idea about the effect of all above aspects on our experiment. Therefore, we can have better confidence in our results. As we are building the model at present time, we've already taken a few of these quality assurance measures, such as hyperparameter fine-tuning. Yet, due to time constraints, we have no choice but to leave the rest of quality assurances to future work.

2.9 Question 9

How does your experimental design ensure repeatability and reproducibility?

We have documented in detail about our experiment process and results in this report. We have open sourced our entire codebase on Github [14]. It includes our implementation of capacity requirement estimation algorithm, capacity progression plotting, our pipeline for running/training models with different number of epochs / percentage of data / model MEC, and all the other codes that generate the results in this report. The output from *Brainome* and *TFMeter* are also included. More importantly, we have integrated *Weights and Bias* platform to track and log the model specifications, hyperparameters and performance of every single run of our experiments. The results can be accessed at [15]. A screenshot of the experiment logs is shown in Figure 10.

Finally, the *Fashion-MNIST* is an open and accessible dataset on which everyone can easily run our codebase and reproduce our results.

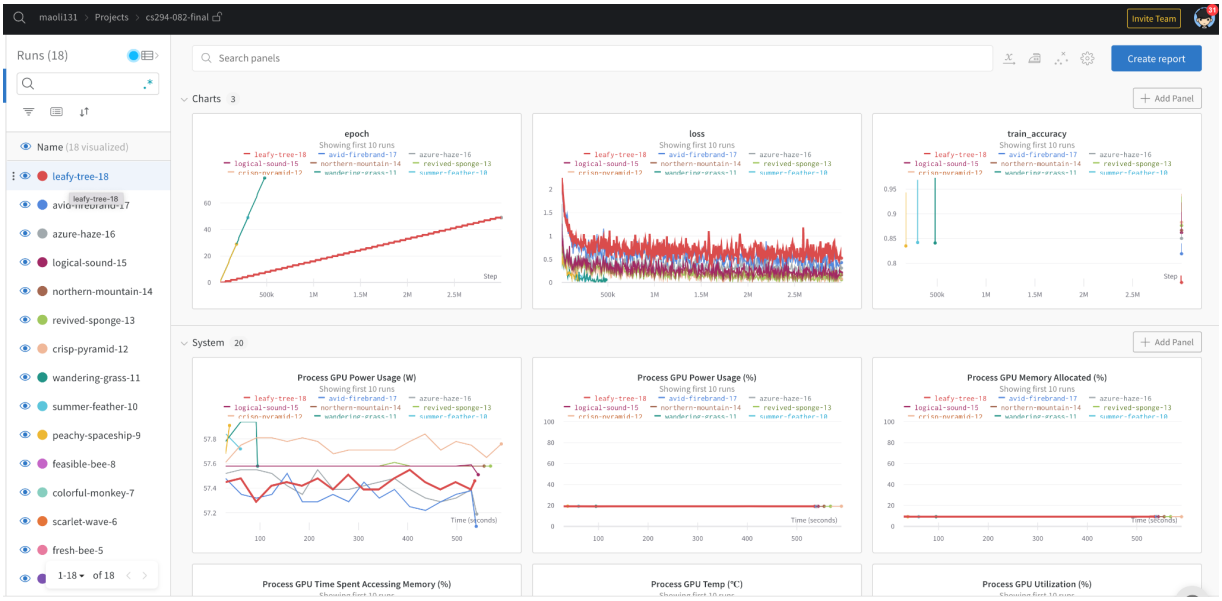

Figure 10. Experiment Logs Screenshot on Weights and Bias Platform

3 ACKNOWLEDGEMENT

Mao Li is the author of our open-sourced codebase in [15]. He implemented the capacity estimation algorithm and he setted up the pipeline for running our experiments in *PyTorch* and logging our results in *Weights and Bias*. Mao is also the main contributor to our project writeup. He drafted the abstract, setup and question sections of this writeup. Cheng Guo comes out with the paper we are referring to, to help transfer the topic into our project design. During the experiment, he is mainly in charge of adversarial sample design and simulation using *TFmeter*. Also he helps to finish and format the final paper. Kexin Guo is in charge of running experiments with *Brainome*. She trained our machine learning model at different Memory Equivalent Capacity and measured how it performed on our dataset. Kexin also helped brainstorming, organizing and formatting the final report.

Last but not least, special thanks to prof. Gerald Friedland for all the lectures, office hours, and guidance that made this work possible. Thanks to our amazing TA Eleanor Cathon for their discussion sections, feedback and assistance.

4 REFERENCE

1. Rosenblatt, Frank. *Perceptions and the theory of brain mechanisms*. Spartan books, 1962.
2. Friedland, Gerald. *Experimental Design for Machine Learning on Multimedia Data.* https://www1.icsi.berkeley.edu/~fractor/fall2021/, 2021
3. *Brainome*. https://www.brainome.ai/.
4. Henrik, Høiness. *Tensorflow Meter*. http://tfmeter.icsi.berkeley.edu, March 16th, 2020.
5. *Weights and Bias.* wandb.ai.
6. Xiao, Han, Kashif Rasul, and Roland Vollgraf. *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.* arXiv preprint arXiv:1708.07747, 2017.
7. Agarwal, Rahul. *The 5 Classification Evaluation Metrics.* Toward Data Science, https://towardsdatascience.com/the-5-classification-evaluation-metrics-you-must-know-aa97784ff226, September, 2019.
8. Piazza Post. *Final report questions.* https://piazza.com/class/kss3z4d4x027ag?cid=59, December, 2021.
9. Piazza Post. *Final report questions.* https://piazza.com/class/kss3z4d4x027ag?cid=59, December, 2021.
10. Friedland, Gerald. Lecture 7 - 8. Book Chapter 9 *An Information View on Data Science*. https://cdn-uploads.piazza.com/paste/jeqdgp7ec8fv4/bb8d09923f28f8df1326688d66f267909167aa329fce3b0f33616c54e562338e/Information_View_on_Data_Science.pdf
11. Piazza Post. *Project FAQ.* https://piazza.com/class/kss3z4d4x027ag?cid=90, December, 2021.
12. Li, Mao. Capacity Requirement Estimate Notebook. Github. December 15th, 2021. https://github.com/maoli131/experimental-design/blob/main/CapacityReq.ipynb
13. Friedland, Gerald. Neural Network MEC computation. *A practical approach to sizing neural networks*. https://arxiv.org/pdf/1810.02328.pdf
14. Li, Mao. Experimental Design Project Repo. Github. December 15th, 2021. https://github.com/maoli131/experimental-design
15. Li, Mao. Weights and Bias Workspace for Experiments Logs, CS294-082. December 15th, 2021. https://wandb.ai/maoli131/cs294-082-final/
16. Cawthon, Eleanor. Section 8 and Section 9, Slides and Recoding. https://bcourses.berkeley.edu/courses/1510267/external_tools/70734
17. Cawthon, Eleanor. Dataset MEC correction. Piazza Post 92. https://piazza.com/class/kss3z4d4x027ag?cid=92