CS 4375 Project: Image Classification Using Existing and New CNN Architectures

Sarisha Ahmed
sxa210172
Department of Computer Science
sxa210172@utdallas.edu

Yifan Ren
yxr170005
Department of Computer Science
yxr170005@utdallas.edu

Abstract (Sarisha: 10% Yifan: 90%)— This studies and compares the performance of Convolutional Neural Network (CNN) over two well-known image datasets: MNIST and Fashion-MNIST. We present a baseline architecture and modified CNN architecture for external investigation of class separability, analyzing how well the model generalizes across models of varying visual complexity. We track the behavior of these models by evaluating the training loss and validation accuracy across epochs. In addition, we adopt reinforcement learning (RL)-style reward tracking for visualization of learning performance and employ both linear and non-linear data-dimension reduction methods including LDA, PCA, and t-SNE to examine dataset class separability.

1. Introduction (Sarisha: 100% Yifan: 0%)

The primary focus of this project is to gain a more thorough understanding of CNN architecture design as well as performance trade-offs. The effectiveness of CNNs lies in their ability to capture and extract relevant patterns from image pixels through convolutional layers, making them a key contributor in the advancement of computer vision technologies. Research behind the implementation of CNNs translates to real world image recognition problems, transforming how computers interpret visual data across all industries.

The motivation behind this study is to compare how model architectures can adapt to datasets of various complexity. We aim to analyze the ways in which CNN architectures generalize, and the complexities surrounding classifying more abstract data. We also intend to demonstrate how dimensionality reduction provides further insight into the model's learning behavior and how the data is structured. These methods include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE). This study particularly highlights a comparison of PCA, a linear dimensionality reduction method that preserves the largest pairwise distances and projects along directions of highest variance, and t-SNE, a nonlinear method that works particularly well for high-dimensional data and preserves local similarities.

Through our analysis of these architectures and datasets, we hope to contribute to a broader understanding of the design and evaluation of models and image classification of simpler complexity.

II. Related Work (Sarisha: 100% Yifan: 0%)

The concept of visualizing and analyzing learned feature representations from convoluted neural networks has long been a challenge for researchers in the field of machine learning. Fundamental work by LeCun et al. [1] introduces gradient-based learning techniques. It established CNNs as a powerful tool that can be used to classify high-dimensional patterns—for the purpose of their work, they compare methods of handwritten character recognition and apply it to document recognition. LeCun et al. [2] and Xiao et al. [3] have since introduced the MNIST and Fashion-MNIST datasets respectively to serve as benchmarks for evaluating the performance of such models. With MNIST gaining popularity due to its convenient size and compatibility with machine learning libraries such as scikit-learn and frameworks such as TensorFlow, Xiao et al. [2] aimed to present a more challenging classification task with Fashion-MNIST while maintaining MNIST's straightforward encoding.

Researchers coupled dimensionality reduction techniques with the complex internal representations of deep learning networks on large datasets by projecting their feature spaces into a lower level of dimensions. Jolliffe [6] details the method of Principal Component Analysis (PCA), making it possible to preserve the variability of the original data. Although simple and effective, the limitation arises of capturing non-linear structures in such complex data. This issue it addressed by van der Maaten and Hinton's [5] introduction of t-SNE, a method touted as the best of those existing at the time for its ability to visualize data

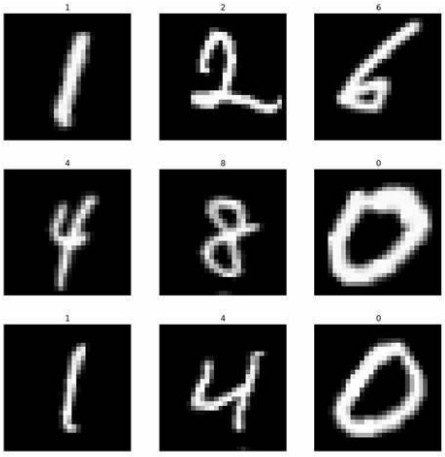
with high-dimensionality, notably images of objects "seen from multiple viewpoints". These methods together give us a comprehensive framework for feature space visualization.

The models we use are trained with the Adam optimizer. Kingma et al. [4] introduced Adam as a gradient-based optimization of stochastic functions, combining the benefits of the Adaptive Gradient Algorithm and Root Mean Square Propagation. With simple implementation, computational efficiency, little memory requirement, and constant in the face of diagonal rescaling of gradients, Adam is popularly adopted today for training in deep learning. For our research we implement the PyTorch framework [8], which conveniently supports GPU acceleration and provides both a convenient way to build neural networks as well as execute Reinfocement Learning (RL) techniques, perfect for our usage.

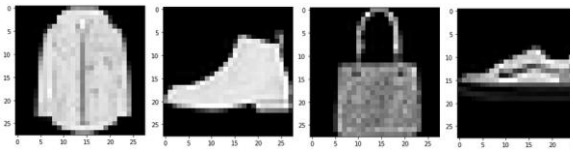
III. DATA SET (SARISHA: 50% YIFAN: 50%)

We use two datasets:

1. MNIST: contains 70,000 grayscale images of handwritten digits (0-9), with 60,000 samples used for training and 10,000 for testing. Each image is 28x28 pixels, and the task is a 10-class classification problem.



2. Fashion-MNIST: Fashion-MNIST mirrors MNIST's structure but features grayscale images of fashion items across 10 categories (e.g., t-shirts, coats, shoes). Despite having the same dimensions and sample count, the dataset presents increased intra-class similarity and inter-class confusion



3. Preprocessing: All images are normalized and converted into PyTorch tensors. The training data is split into 90% training and 10% validation. This uniform preprocessing ensures fair comparison between datasets.

Both datasets consist of 28 * 28 grayscale images with 60,000 training samples and 10,000 testing samples.

IV. Model Architecture and Training

Both models accept a 28×28 grayscale image as input (1 input channel). Below we describe their architectural differences.

- LeNet:
  - Convolution Layers: 2 layers (6 to 16 channels)
  - Kernel Size: 5 * 5
  - Activation Function: Tahn
  - Pooling: Average pooling
  - Dropout: none
  - Fully Connected: 16 * 4 * 4 to 120 to 84 to 10
  - Regularization: None

- Custom CNN
  - Convolution Layers: 2 Layers (32 to 64 channels)
  - Kernel size: 3 * 3 with padding = 1
  - Activation Function: ReLu
  - Pooling: Max pooling (MaxPool2d)
  - Dropout: 0.25 after convolutional block, 0.5 after first fully connected layer.
  - Fully Connected: 64 * 14 * 14 to 128 to 10
  - Regularization: Dropout
- Training Settings
  - Optimeter: Adam
  - Loss Function: CrossEntropyLoss
  - Batch size: 64
  - Epochs: 30
  - Validation split: 10% of training set

All models are trained using PyTorch on a CUDA-enabled GPU when available

2. V. RESULT AND EVALUATION (SARISHA: 50% YIFAN: 50%)

- Accuracy Comparison:

Model	Dataset	Validation Accuracy%
LetNet	MNIST	98~
Custom CNN	MNIST	99.3~
LetNet	Fashion-MNIST	89~
Custom CNN	Fashion-MNIST	93.47~

The CustomCNN architecture consistently outperforms LeNet in terms of classification accuracy, particularly on the more complex and visually diverse Fashion-MNIST dataset. This performance gain can be attributed to the use of deeper convolutional layers, ReLU activations, and dropout regularization, all of which contribute to better generalization and feature discrimination. While both models perform well on the simpler MNIST dataset, the CustomCNN demonstrates a more robust ability to capture nuanced patterns and subtle

inter-class differences in Fashion-MNIST, leading to higher validation accuracy and more clearly separated class clusters in LDA, PCA, and t-SNE visualizations.

However, this improved performance comes at a computational cost. Training the CustomCNN requires noticeably more time compared to LeNet, primarily due to its larger number of parameters and more complex architecture. For instance, training CustomCNN for 30 epochs takes approximately 25–30 minutes on a standard CPU-enabled environment, whereas LeNet completes in significantly less time under the same conditions. This trade-off between accuracy and efficiency highlights a key consideration when selecting models for real-world applications where computational resources or inference latency may be constrained.

Fig for LetNet/MNIST:

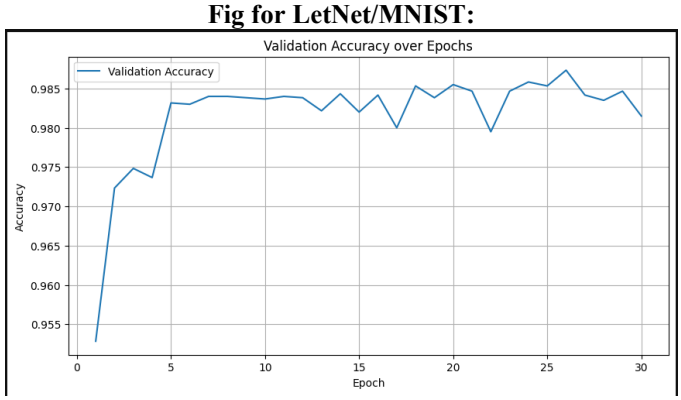


Fig for LetNet/Fashion-MNIST:

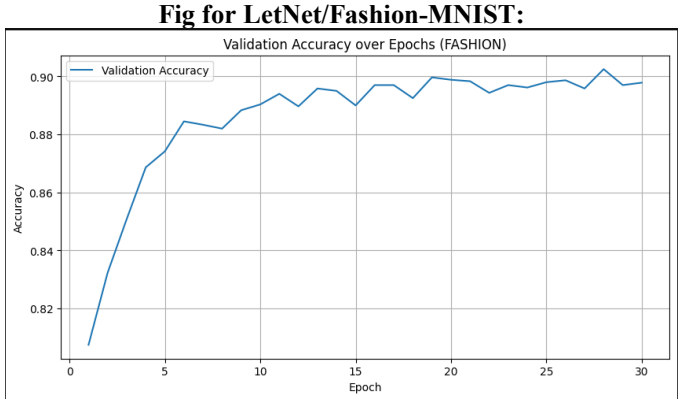


Fig for CustomCNN/MNIST:

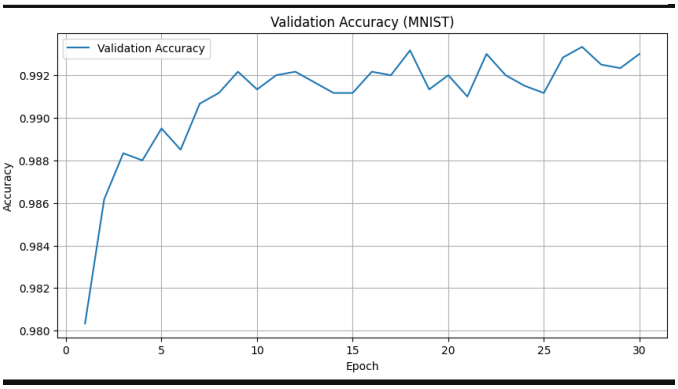
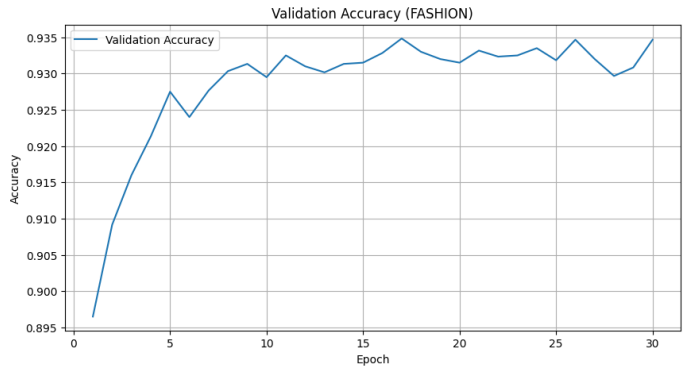
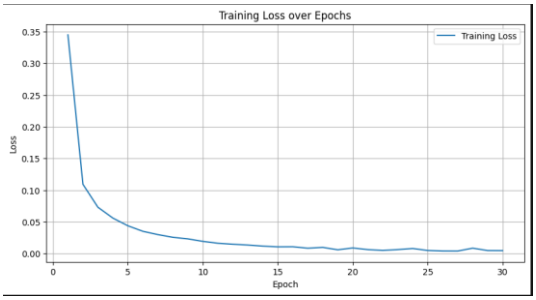


Fig for CustomCNN/Fashion-MNIST:

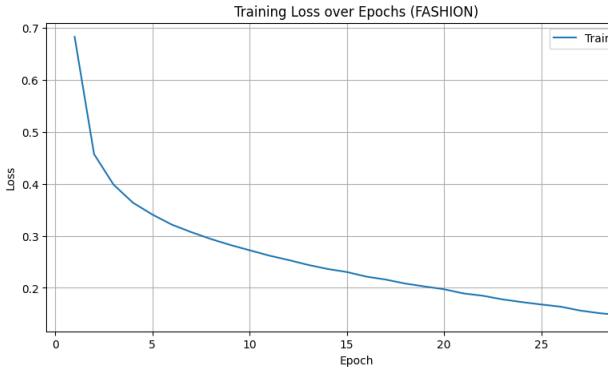


- **Loss Analysis for LeNet on MNIST**:
  - LeNet on MNIST
  - **Initial Loss**: ~0.3317
  - **Final Loss**: ~0.0110
  - **Trend**: Steady and smooth decrease over 30 epochs.
  - **Insight**: Despite being an older architecture, LeNet trains effectively on MNIST, achieving **very low loss**, showing that it's still highly suitable for simple digit recognition tasks.

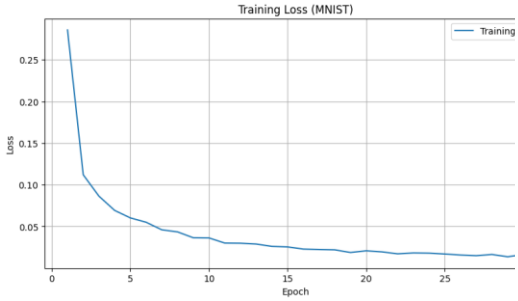


- **Loss Analysis for LetNet on Fashion-MNIST:**
  - **Initial Loss**: ~0.6831
  - **Final Loss**: ~0.1432
  - **Trend**: Steady and smooth decrease over 30 epochs.

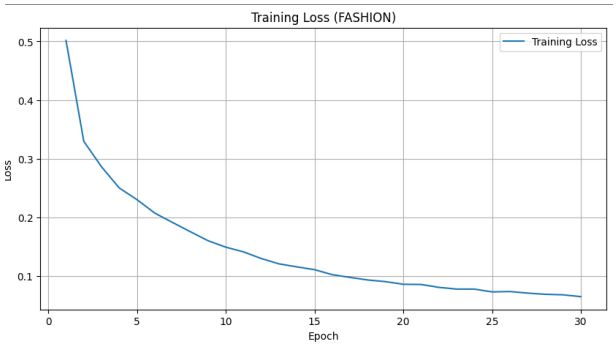
- **Insight**: Despite being an older architecture, LeNet trains effectively on Fashion MNIST, achieving **very low loss**, showing that it's still highly suitable for simple digit recognition tasks.



- **Loss Analysis for CustomCNN on MNIST:**
  - **Initial Loss**: ~0.2858
  - **Final Loss**: ~0.0163
  - **Trend**: Smooth decrease but final loss slightly higher than LeNet.
  - **Insight**: CustomCNN also converges well, and faster in early epochs. However, **LeNet achieves slightly lower final loss**, suggesting better overfitting control or a more optimal architecture for MNIST in this case.



- **Loss Analysis for CustomCNN on Fashion-MNIST:**
  - **Initial Loss**: ~0.5015
  - **Final Loss**: ~0.0652
  - **Trend**: Steady decrease, but much higher than MNIST results.
  - **Insight**: Fashion-MNIST is more complex, and even the stronger CustomCNN struggles more, but still reaches a reasonably low loss, confirming decent generalization.



VI.FEATURE VISUALIZATION(S: 50% Y: 50%)

Dimensionality reduction techniques are applied to the final layer embeddings from the trained models.

- PCA

Principal Component Analysis was used to reduce features to 2D for visualization. MNIST classes are mostly well-separated; Fashion-MNIST classes overlap more.

Fig. 1 shows a PCA projection of LeNet's extracted features on the MNIST test set. Each point represents an image, colored by its true digit label (0–9). Most digits form distinct clusters, such as '0', '1', and '9', indicating good feature separability. However, overlaps between some classes, like '4', '7', and '9', suggest that these digits are harder to distinguish. This visualization helps assess how well the model differentiates between classes and highlights areas for improvement.

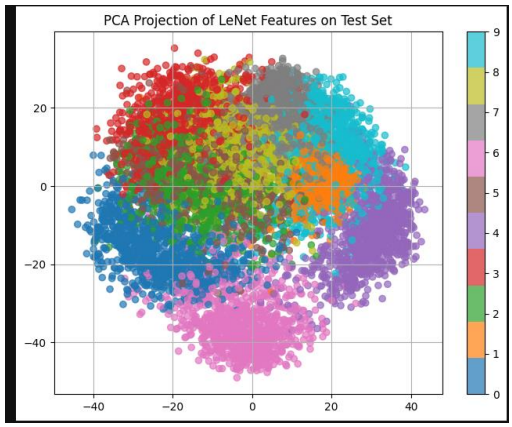


Fig. 2 displays the PCA projection of MNIST features from the CustomCNN model. While some classes like '1' and '0' are relatively well-separated, there is more noticeable overlap among digits such as '3', '5', and '8'. This suggests that CustomCNN's learned features are slightly less discriminative compared to LeNet in this projection, indicating potential room for improvement in class separation.

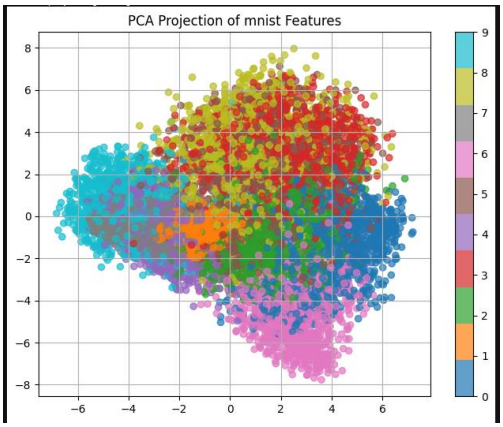
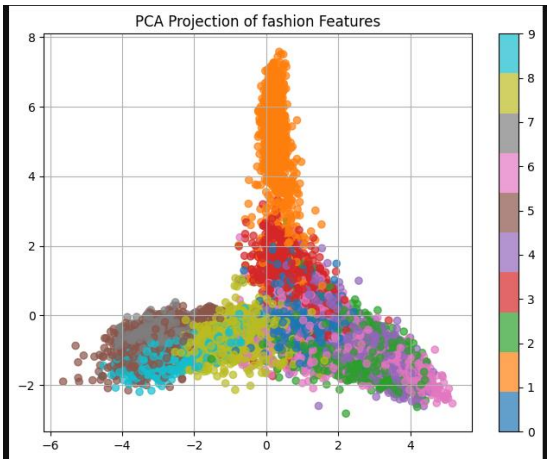


Fig. 3 shows the PCA projection of features from the Fashion-MNIST dataset. Unlike MNIST, the clusters here are less compact and exhibit significant overlap, especially around the center. Notably, class '1' (orange) forms a very tight vertical cluster, while others such as '2', '3', and '5' are harder to distinguish. This suggests that Fashion-MNIST is more challenging to separate in the feature space and may benefit from more advanced architectures or nonlinear projection methods like t-SNE for clearer visualization.



- t-SNE
t-SNE was used for nonlinear dimensionality reduction. While visually appealing, t-SNE plots showed entanglement for certain Fashion-MNIST classes (e.g., Shirt vs. Coat).

Fig. 4 displays the t-SNE projection of LeNet-extracted features from the test set. Compared to PCA, t-SNE reveals clearly separated and well-defined clusters for each digit class. This highlights t-SNE's strength in capturing complex, nonlinear relationships in the data. The distinct cluster boundaries suggest that LeNet effectively encodes class-specific information in its learned representations.

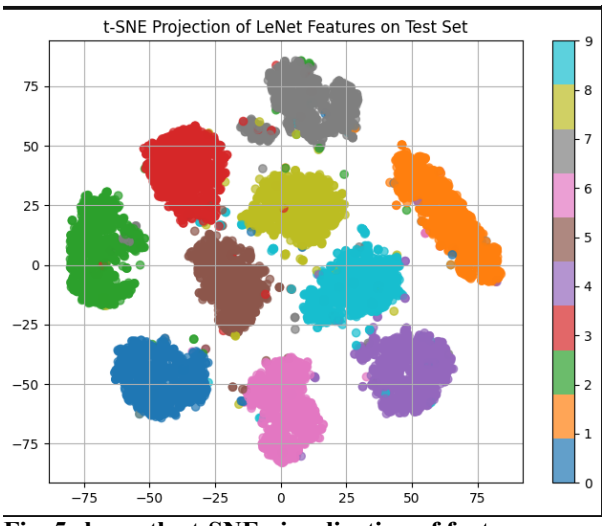


Fig. 5 shows the t-SNE visualization of features extracted from MNIST. The digit classes are well separated into distinct clusters, indicating strong feature learning by the model. Some overlap is still visible (e.g., between digits like 4 and 9), reflecting common handwriting ambiguities. Overall, t-SNE effectively reveals the underlying structure in MNIST embeddings.

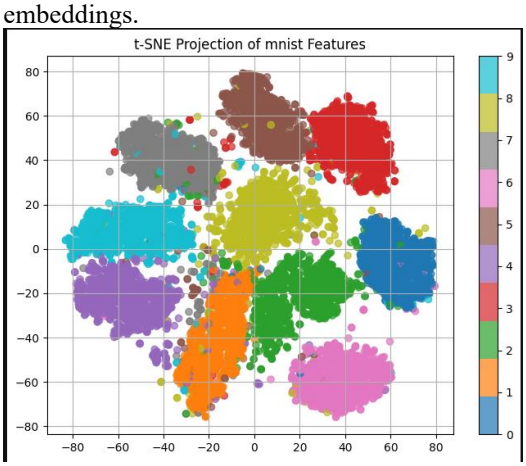
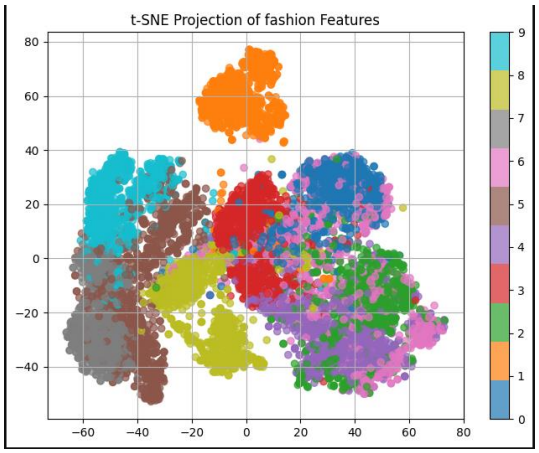


Fig. 6 shows the t-SNE visualization for Fashion-MNIST. Compared to MNIST, the clusters are more overlapping, especially among classes like T-shirt, shirt, and pullover, indicating greater similarity and higher classification difficulty. Nonetheless, certain categories (e.g., sneakers and coats) remain well separated, suggesting distinctive features for those classes.



VII. DISCUSSION (SARISHA: 50% YIFAN: 50%)

Our results suggest that the deeper, dropout-regularized networks like CustomCNN perform marginally better in both accuracy and generalization as compared to the traditional architecture (LeNet), especially on complex dataset like Fashion-MNIST. Although the final validation loss for CustomCNN is slightly more than LeNet on MNIST, its accuracy is slightly outperformed. This implies that the wider model is well generalized.

The ReLU and max pooling used in CustomCNN are responsible for more effective gradient back propagation and better feature preservation than LeNet's tanh and average pooling. Feature visualization using PCA and t-SNE also corroborates that CustomCNN learns more separable and discriminative features across the classes.

But these advantages are achieved at the risk of not being computationally efficient. Because of its depth and regularization techniques such as dropout, it takes around one hour for CustomCNN to train whereas LeNet takes only a few minutes. LeNet could still be a good alternative in some circumstances where timing is critical or resources are scarce, thanks to its relatively quick training process and surprising success on simpler datasets, such as MNIST.

3. *VIII. CONCLUSION* (SARISHA: 0% YIFAN: 100%)

In this work, we compared two CNN models (the standard LeNet, and a modern Dropout-regularized CustomCNN) on two standard datasets, MNIST and Fashion-MNIST. The objective was to study and compare their performance in the methodologies of training-behavior, classification-accuracy, feature-separability and visual representations by dimensionality reduction using PCA and t-SNE.

Our findings indicate that although LeNet is still a relatively lightweight architecture and efficient for simpler tasks such as digit classification, CustomCNN constantly outperformed it in terms of classification accuracy and feature discriminativeness, especially on the more challenging Fashion-MNIST dataset. Improvements in accuracy were not very significant on MNIST but more on Fashion-MNIST, hinting that deeper models with improved regularization do provide a clear benefit when using a dataset which has higher inter-class variation or and overlapping features.

These results were confirmed by the feature projection — the learnt 3D visual representations provided significantly more separated and compact class distributions in 2D feature space, suggesting superior feature learning ability of CustomCNN. These gains come from contemporary architectural decisions - ReLU activations for excellent gradient flow, max pooling for increased spatial abstraction, and dropout for both regularization and enhanced generalization.

However, there are trade-offs to these gains. The CustomCNN model suffered from much longer training times, probably because of its higher depth and regularization layers. In comparison, LeNet (with millions of less parameters) was faster to train and still top-performs on MNIST, showing that simpler models can work quite well for some tasks if the computational restrictions are sufficient.

Finally, our results illustrate that state-of-the-art deep learning architectures can be very beneficial with respect to enhanced model performance and generalization, in particular for challenging datasets. On the other hand, architectural decisions should always be confined by the computation budget and the nature of the target task. Possible avenues of future research would be to use other backbones, consider transfer learning, or try to apply these models to real-world, imbalanced and/or noisy datasets.

4. *IX. SETUP* (SARISHA: 50% YIFAN: 50%)

All experiments were conducted on a personal desktop machine with the following specifications:

- Device Name: DESKTOP-DGOCFDK
- Processor: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
- RAM: 16.0 GB
- System Type: 64-bit Operating System, x64-based Processor
- GPU: No discrete GPU was used; all computations were performed on CPU
- Operating System: Windows 10 Home

Each model (LeNet and CustomCNN) was trained for 30 epochs on both the MNIST and Fashion-MNIST datasets. The training and validation splits were set to 90% and 10% respectively. A batch size of 64 was used throughout. The optimizer was Adam with a learning rate of 0.001, and CrossEntropyLoss was used as the loss function.

Due to the absence of a dedicated GPU, training was performed using CPU computation only. Consequently, LeNet completed training significantly faster than CustomCNN, which incurred longer training times due to its more complex architecture and larger parameter count.

X: Visualization Methods and Justification (Sarisha: 50% Yifan: 50%)

To better understand the learned feature representations and the separability of classes in the latent space, we applied three widely used dimensionality reduction techniques: Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-distributed Stochastic Neighbor Embedding (t-SNE).

1. PCA (Principal Component Analysis)

PCA is a linear method that projects data onto directions (principal components) that maximize variance. It is computationally efficient and easy to interpret. However, since it does not consider class labels, PCA might not clearly separate classes if their distributions overlap significantly. The following provides a simple standard formulation of the given covariance matrix and the transformation of the *P*-dimensional data in a 2-dimensional space:

$$\hat{\Sigma} = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}x^{(i)^T} = \frac{1}{N}X^T X$$

$$Z = XW$$

Given:

$$X = \left[x^{(1)}, \cdots, x^{(N)}\right]^T \in \mathbb{R}^{N\times P}$$

2. t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE is a non-linear, unsupervised method that is particularly effective at visualizing high-dimensional data in 2D or 3D. It preserves local structure and is excellent for observing clusters and neighborhood relationships. However, it is computationally intensive, can be sensitive to hyperparameters like perplexity, and does not preserve global distances well. Given high-dimensional data points {x1 ,x2 ,…,xn}, t-SNE computes pairwise similarities using conditional probabilities, then models them in the low-dimensional space using Student t-distribution:

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|x_i - x_k\|^2/2\sigma_i^2\right)}$$

$$q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}}$$

XI. FUTURE WORK (Sarisha: 50% Yifan: 50%)

Although this work has successfully compared LeNet and a custom CNN for MNIST and Fashion-MNIST datasets, there are still a number of exciting areas to explore for future work. First, we will investigate more complex and deep architectures such as ResNet or VGG to assess how deeper backbones influence the performance and training times in small-scale datasets. Second, using some techniques including data augmentation (e.g., rotation, translation, random erasing) may further improve the generalization, especially in the case of Fashion-MNIST, where inter-class similarity is high.

Furthermore, ideas for future research might include hyperparameter tuning (e.g., learning rate schedules, batch sizes, dropout rates), and experimenting with variations of optimizers, such as SGD with momentum or RMSProp, to enhance speed of convergence. On the visualization front, incorporation of more sophisticated class separation tools such as UMAP or SHAP might offer greater clarity regarding the distribution of features and the interpretability of the model.

Finally, deployment and evaluation on real-world noisy data or edge devices would also offer a realistic view of model robustness and efficiency other than benchmark datasets.

References

[1] *Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.*

[2] Y. LeCun, C. Cortes, and C. J. Burges, "The MNIST Database of Handwritten Digits," [Online]. Available: http://yann.lecun.com/exdb/mnist/

[3] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," arXiv preprint arXiv:1708.07747, 2017.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[5] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," Journal of Machine Learning Research, vol. 9, pp. 2579–2605, 2008.

[6] I. T. Jolliffe, Principal Component Analysis, 2nd ed., Springer, 2002.

[7] R. A. Fisher, "The use of multiple measurements in taxonomic problems," Annals of Eugenics, vol. 7, no. 2, pp. 179–188, 1936.

[8] PyTorch Team, "PyTorch: An open source machine learning framework," [Online]. Available: https://pytorch.org/