RAS598, Fall 2024

Fashion MNIST Classification with Deep CNN and Other Machine Learning Models

KARAN ATHREY, ABHIJIT SINHA AND ANUSHA CHATTERJEE

Arizona State University

Contacts: kathrey@asu.edu , asinh117@asu.edu , and achatt53@asu.edu

INTRODUCTION

Context:

The project exemplifies the transformative potential of machine learning in organizational contexts, highlighting opportunities for operational excellence, innovation, and sustained competitive advantage.

Objective:

- In this project, we will develop a comprehensive machine learning pipeline to classify images from the Fashion MNIST dataset.
- The primary goal is to implement a deep Convolutional Neural Network (CNN) using TensorFlow and compare its performance with other traditional machine learning algorithms.
- We will utilize Plotly for interactive visualizations to better understand the data and the performance of our models.

Significance:

By implementing advanced machine learning techniques to classify fashion items, organizations can enhance operational efficiency, improve customer experience, and gain valuable insights for strategic decision-making.

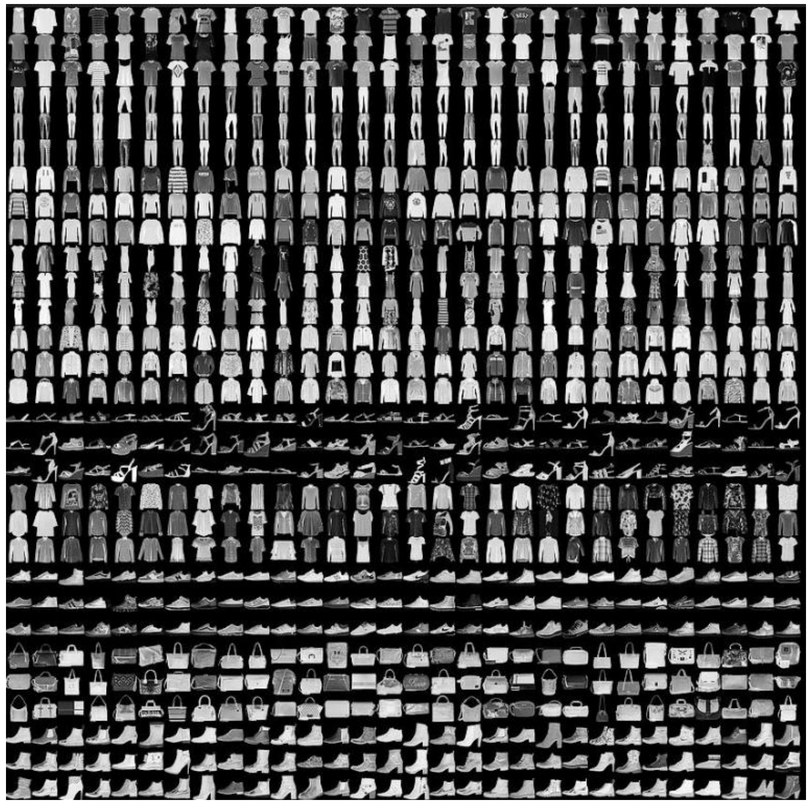


Fig: Fashion MNIST dataset

Problem Statement

Real-World Examples

- **Amazon and E-commerce Platforms**: Utilize image classification for product recommendations, search optimization, and inventory management.
- **Fashion Retailers**: Companies like Zara and H&M use AI to analyze trends and customer preferences, informing design and stocking decisions.
- **Social Media Platforms**: Pinterest and Instagram employ image recognition to enhance user experience through features like shoppable posts.
- **Manufacturing**: Brands use image classification for quality control on production lines, ensuring consistency and reducing defects.

Key Questions:

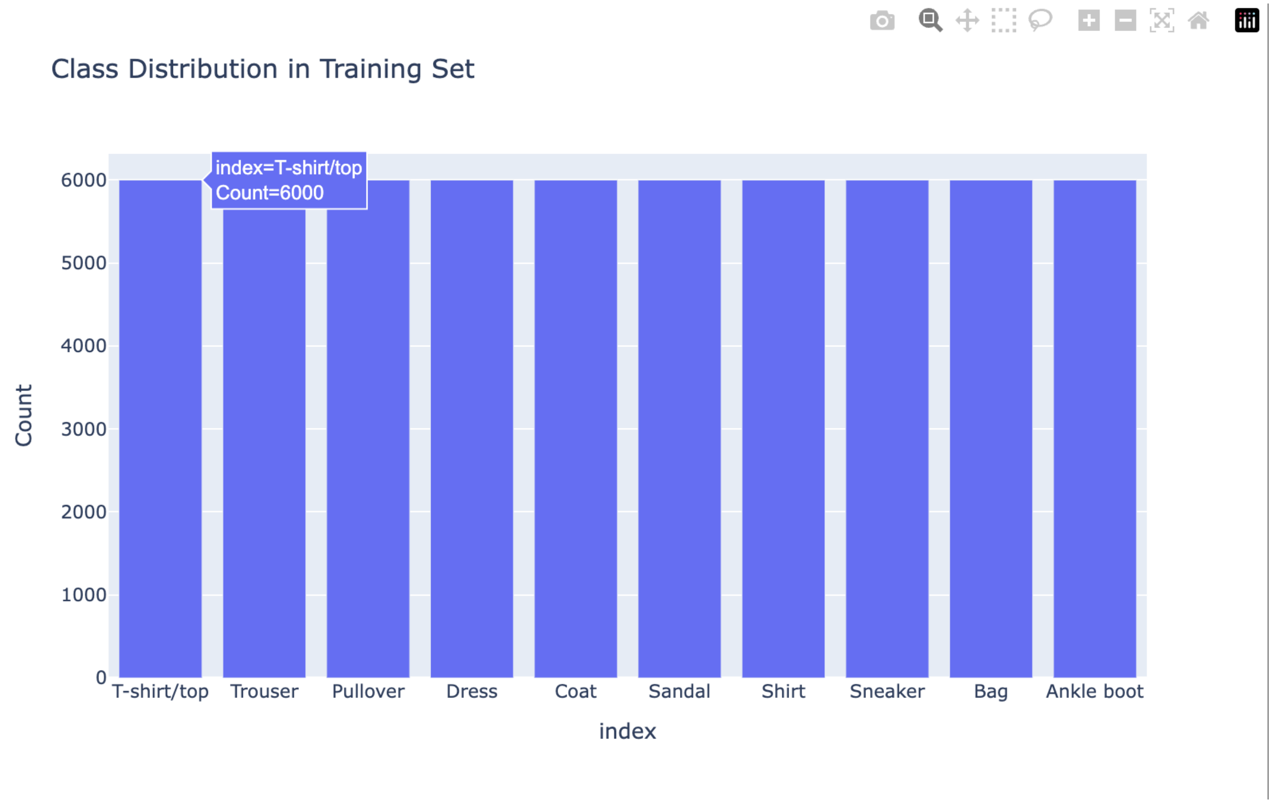
- How does the performance of a deep convolutional neural network (CNN) compare to other machine learning models in classifying Fashion MNIST images?
- What techniques can be employed to optimize machine learning models for better accuracy on the Fashion MNIST dataset?

Data Description

Dataset Overview:

The Fashion MNIST dataset is a drop-in replacement for the original MNIST dataset of handwritten digits. It consists of 70,000 grayscale images of fashion products, each of size 28x28 pixels, belonging to 10 categories:

1. T-shirt/top
2. Trouser
3. Pullover
4. Dress
5. Coat
6. Sandal
7. Shirt
8. Sneaker
9. Bag
10. Ankle boot



The dataset is split into 60,000 training images and 10,000 test images.

Data Analysis

Fashion MNIST Dataset Overview

1. Dataset Composition:

- **Training Set**: 60,000 images of size 28×2828 \times 2828×28.
- **Test Set**: 10,000 images of size 28×2828 \times 2828×28.
- **Data Type**: Grayscale images (uint8).
- **Pixel Intensity Range**: 0 (black) to 255 (white).

2. Sample Data Visualization:

- Shows a variety of fashion categories: T-shirts, shoes, trousers, dresses, coats, and bags.
- Labels correspond accurately to the displayed items.

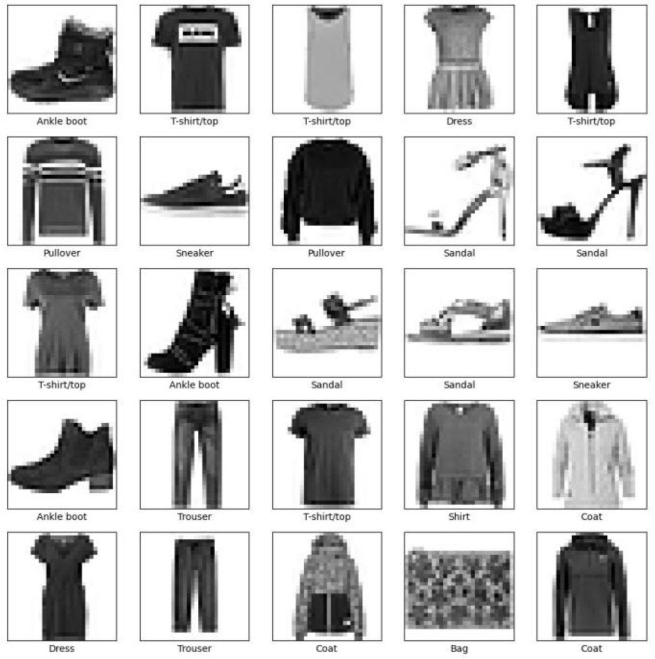
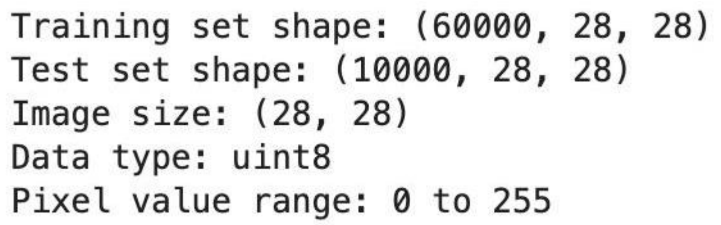


Fig: The given dataset

Data Analysis



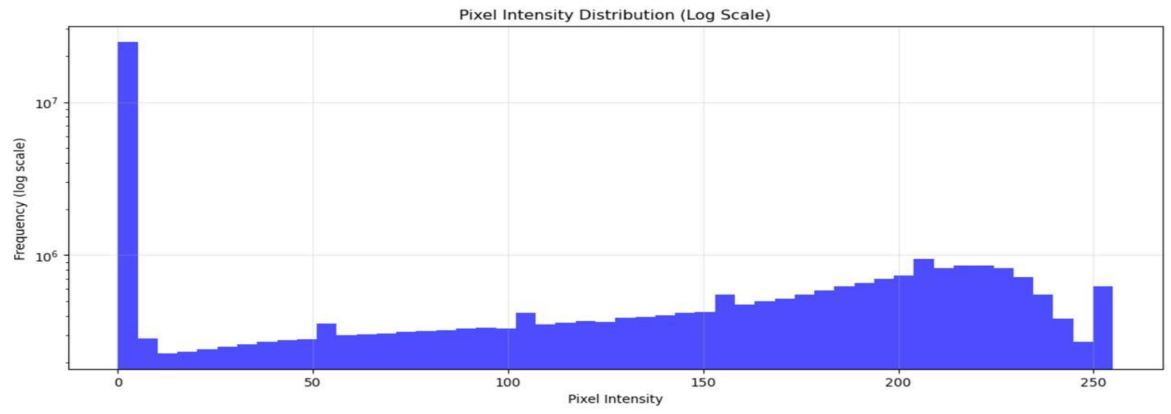
```
Training set shape: (60000, 28, 28)
Test set shape: (10000, 28, 28)
Image size: (28, 28)
Data type: uint8
Pixel value range: 0 to 255
```

Pixel Intensity Analysis

1. Distribution Insights:

- **Pixel Intensity Histogram**:
  - Shows the frequency of pixel intensities on a logarithmic scale.
  - Skewed distribution with most pixel values near 0 (black regions in the image).
- **Statistics**:
  - **Mean Pixel Intensity**: 72.94.
  - **Median Pixel Intensity**: 0 (indicating sparse pixel usage).

Standard Deviation: 90.02.



```
Mean pixel intensity: 72.94
Median pixel intensity: 0.00
Standard deviation: 90.02
```

2. Implications:

- Sparse usage of pixel intensities requires normalization for effective model training.
- Highlights the importance of pre-processing to bring pixel values into a consistent range for ML models.

Preprocessing Steps

- **Data Loading**
- **Step: Load the Fashion MNIST dataset.**
- **Purpose: Ensure the data is accessible for training and testing.**
- **Data Inspection**
- **Step: Visualize a few samples from the dataset.**
- **Purpose: Understand the nature of the data and verify its correctness.**
- **Normalization**
- **Step: Normalize pixel values to a range of `[0, 1]` by dividing by 255.**
- Purpose: Helps in faster convergence during training by ensuring uniform scaling of input features.
- **Data Reshaping**
- **Step**: Reshape the images into a format suitable for the models (e.g., `(28, 28,1)` for CNNs).
- **Purpose: Match the input dimensions expected by deep learning models.**
- **Data Splitting**
- **Step: Split the dataset into training, validation, and test sets.**
- **Purpose: Separate data for training and evaluating model performance.**
- **Flattening (For Non-CNN Models)**
- **Step: Flatten the 2D image arrays into 1D vectors for traditional ML models.**
- **Purpose: Make the data compatible with algorithms like SVM or Logistic Regression.**

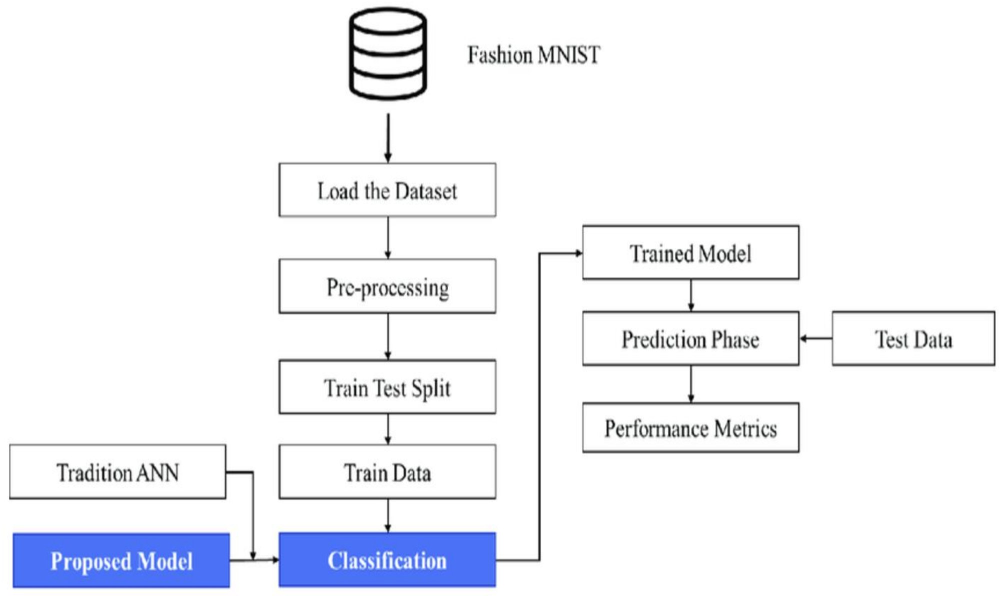


Fig: MNIST fashion dataset processing steps

ML Models Used in This project

- Residual Network (ResNet)
- Convolutional Neural Network(CNN)
- Logistic Regression
- Support Vector Machine(SVM)
- Random Forest
- K-nearest Neighbors(KNN)
- Multi-Layer Perceptron(MLP)

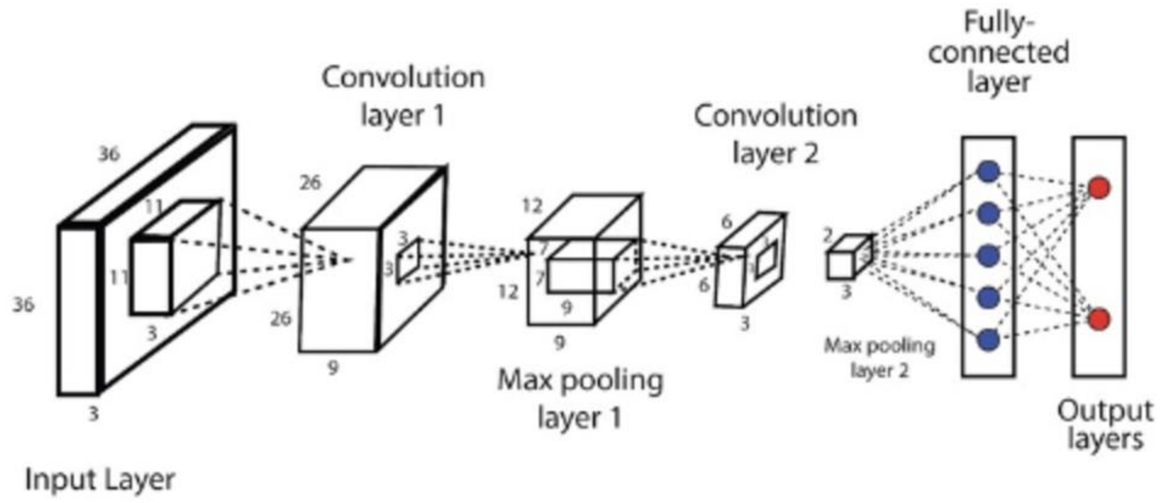


Fig: CNN model

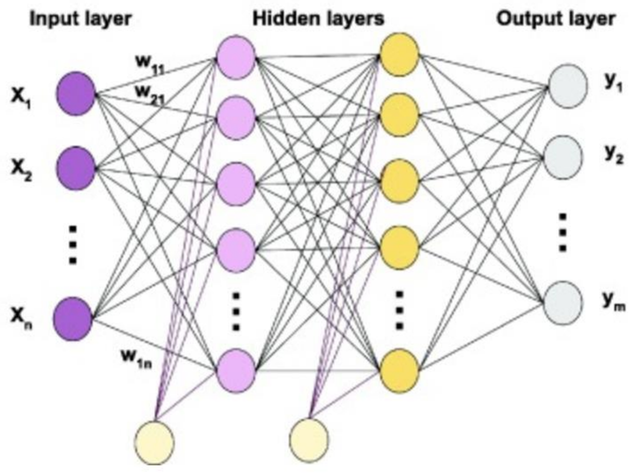


Fig: MLP model

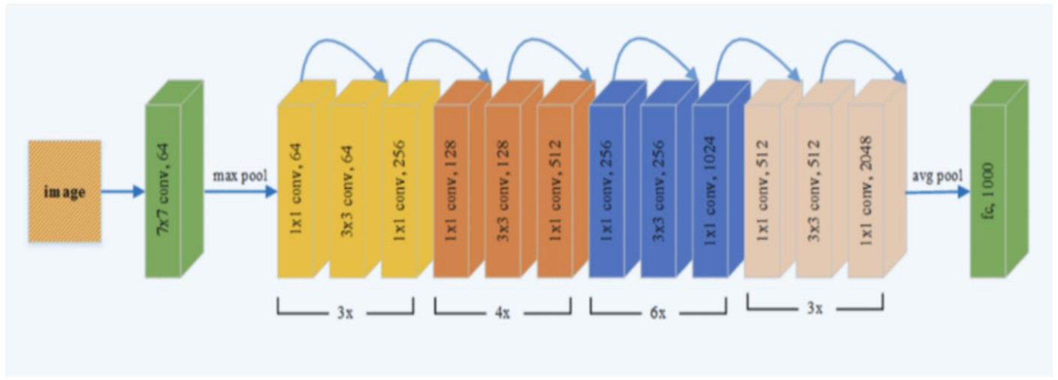


Fig: ResNet Model

ResNet Model

Why ResNet?
ResNet's residual blocks tackle vanishing gradients, enabling deep architectures that capture complex patterns, making it ideal for Fashion MNIST's 10-class classification.

Key Features:

- **Residual Blocks**: Skip connections ensure efficient gradient flow.
- **Batch Normalization**: Stabilizes training.
- **Global Average Pooling**: Reduces overfitting and computational complexity.

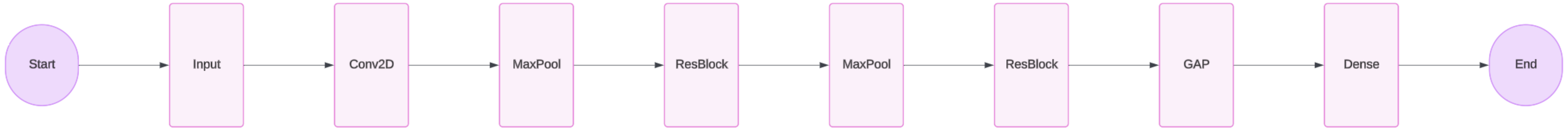
Steps for Implementation:

- **Preprocessing**: Normalize pixel values, reshape to (28,28,1)(28, 28, 1)(28,28,1), and one-hot encode labels.
- **Model**: Use ResNet-18/34 with the final dense layer outputting 10 classes.
- **Training**: Apply data augmentation and train with Adam or SGD optimizer.
- **Evaluation**: Measure accuracy and analyze the confusion matrix.

Advantages:

- Handles deep networks effectively.
- Achieves higher accuracy than traditional CNNs.
- Generalizes well to unseen data.

ResNet Model for Fashion MNIST Classification



Core Building Blocks

Model Characteristics

Feature	Implementation
Filter Size	32 throughout
Convolution	3×3 kernels
Pooling	2×2 MaxPool
Regularization	Batch Normalization
Final Layer	Softmax (10 classes)

Architecture Overview

Input: 28×28×1 grayscale images

Total Parameters: Lightweight implementation

Output: 10 fashion categories

Technical Highlights

- Identity shortcuts for gradient flow
- Global Average Pooling reduces parameters
- Batch normalization for training stability
- Efficient architecture for Fashion MNIST scale

Implementation Benefits

- Lightweight yet effective
- Training stability
- Reduced overfitting risk
- Suitable for fashion classification task

ResNet Model Performance Analysis

Training Metrics

Accuracy Performance

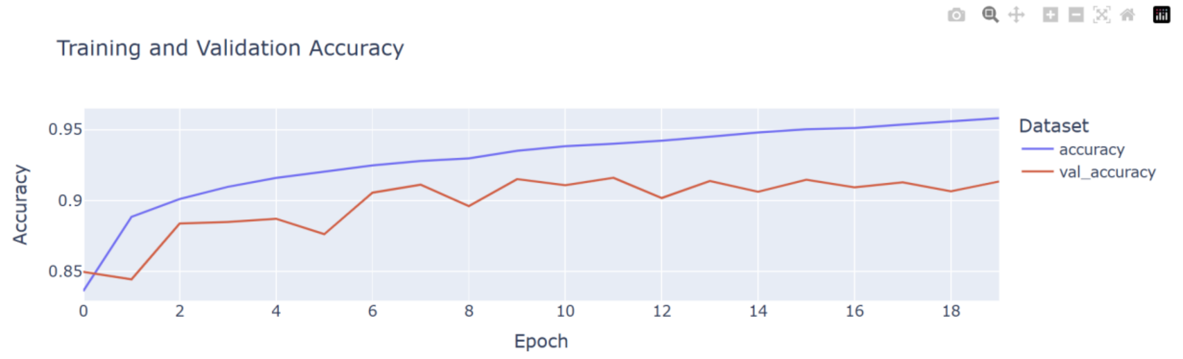
- Training accuracy reaches ~95% after 20 epochs
- Validation accuracy stabilizes at ~91%
- Gap between training and validation (~4%) indicates slight overfitting

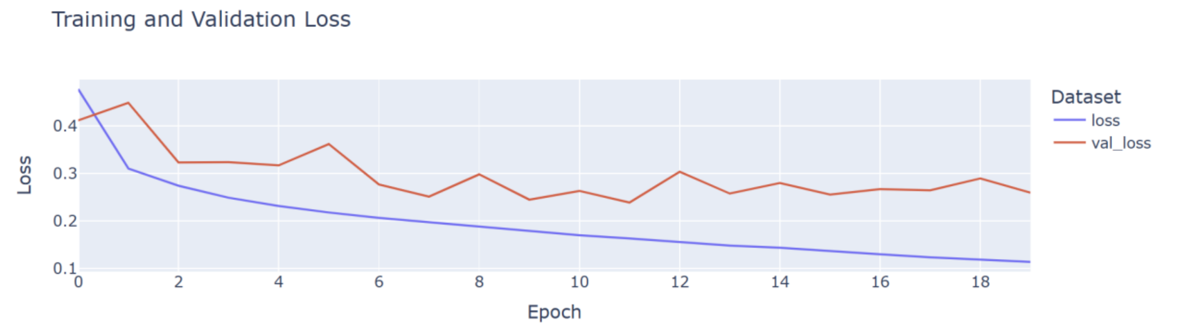
Loss Trends

- Training loss steadily decreases from 0.4 to 0.1
- Validation loss stabilizes around 0.3
- Loss curves suggest model convergence

Key Observations

Metric	Training	Validation
Final Accuracy	95%	91%
Loss	0.1	0.3
Convergence	Steady	Stable





Model Characteristics

- Early Learning: Rapid improvement in first 5 epochs
- Stability: Consistent performance after epoch 10
- Generalization: Good balance between training and validation metrics
- Convergence: No signs of significant overfitting or underfitting

This analysis suggests the ResNet model achieves robust performance on the Fashion MNIST dataset with good generalization capabilities.

ResNet Model Performance Analysis

Confusion Matrix Analysis

Overall Performance

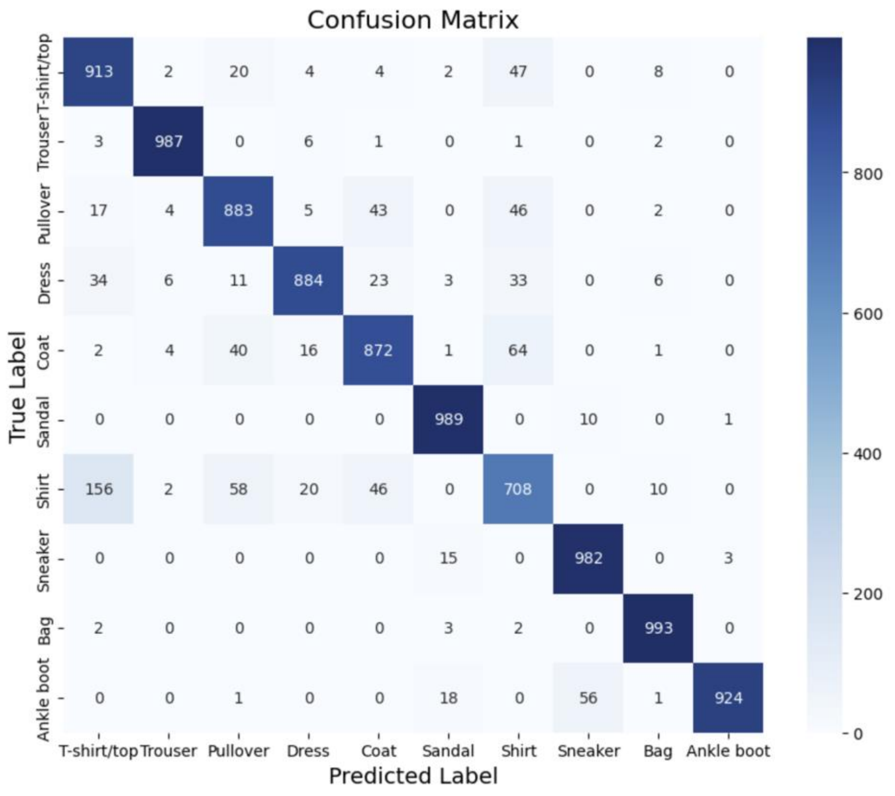
- Model achieves 91% accuracy across all classes
- Strong diagonal dominance indicates good classification

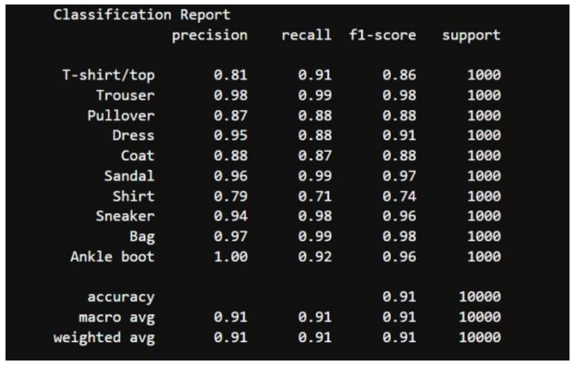
Class-Specific Performance

Category	Key Insights
Best Performers	Bag (99.3%), Sandal (98.9%), Trouser (98.7%)
Challenging Classes	Shirt (70.8%), T-shirt/top (91.3%)

Notable Confusions

- Shirt frequently misclassified as T-shirt/top (156 cases)
- Coat and Pullover show mutual confusion
- Sneaker and Ankle boot have some overlap




```
Classification Report
              precision    recall  f1-score   support

 T-shirt/top       0.81      0.91      0.86      1000
     Trouser       0.98      0.99      0.98      1000
    Pullover       0.87      0.88      0.88      1000
       Dress       0.95      0.88      0.91      1000
        Coat       0.88      0.87      0.88      1000
      Sandal       0.96      0.99      0.97      1000
       Shirt       0.79      0.71      0.74      1000
     Sneaker       0.94      0.98      0.96      1000
         Bag       0.97      0.99      0.98      1000
  Ankle boot       1.00      0.92      0.96      1000

    accuracy                           0.91     10000
   macro avg       0.91      0.91      0.91     10000
weighted avg       0.91      0.91      0.91     10000
```

Performance Metrics

Strong Points
- Most classes show >90% precision
- High recall rates across majority of categories
- F1-scores consistently above 0.85

Areas for Improvement
- Shirt category needs attention (F1: 0.74)
- T-shirt vs Shirt disambiguation
- Outerwear (Coat/Pullover) distinction

This analysis shows robust model performance with specific areas identified for potential enhancement.

Gradient-weighted Class Activation Mapping

What is Grad-CAM?
Grad-CAM (Gradient-weighted Class Activation Mapping) is a popular technique in Convolutional Neural Networks (CNNs) used for visual explanations of predictions made by the model. It is particularly useful for interpreting decisions in image classification and related tasks.

Why Grad-CAM?
- CNNs are often treated as "black boxes.“
- Grad-CAM visualizes the areas of the image that influence model predictions.
- Provides visual explanations for predictions.
- Builds trust in machine learning models for business use.

How Grad-CAM Performed:
- Successfully localized important features (e.g., shape and edges of the boot).
- The heatmap indicates the model relied on key visual cues like the shoe's outline and sole.
- The predicted label matched the true label, showing strong model performance and interpretability.

What it Represents:
- **Model Interpretability**: Explains why the model made its prediction by visualizing important regions.
- **Performance Validation**: Confirms the model focuses on relevant features for classification.
- **Practical Use**: Ensures the model bases predictions on logical and explainable cues rather than irrelevant patterns.

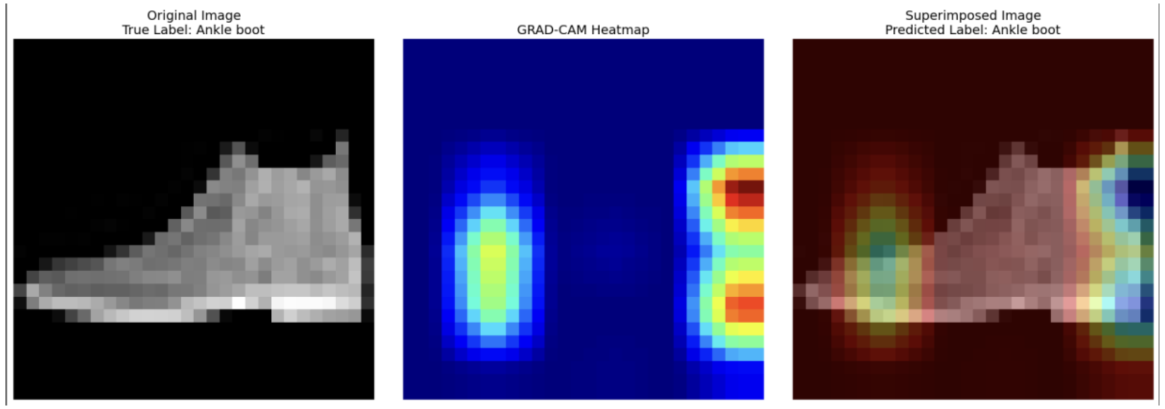


Image Breakdown:
- **Left**: Original Image
- Displays the grayscale input image of the item classified as "Ankle Boot."
- Ground truth label: **Ankle Boot**.
- **Middle**: Grad-CAM Heatmap
- Highlights the regions most influential for the model's prediction.
- Bright areas (yellow/red) indicate where the model focuses to identify the object.
- **Right**: Superimposed Image
- Combines the original image and the Grad-CAM heatmap.
- Shows the model correctly focused on the shoe region for its prediction.

Convolutional Neural Network

Why CNN?
CNNs excel at extracting spatial features, making them ideal for Fashion MNIST's 10-class image classification.

Key Features:
- **Convolutional Layers**: Detect patterns like edges and textures.
- **Pooling Layers**: Reduce spatial dimensions while retaining key features.
- **Fully Connected Layers**: Map features to class predictions.
- **ReLU & Softmax**: Ensure non-linearity and output probabilities.

Steps:
•**Preprocessing**: Normalize pixel values, reshape images (28,28,1)(28, 28, 1)(28,28,1), and one-hot encode labels.
•**Architecture**:
- Conv2D → ReLU → MaxPooling2D → Flatten → Dense → Softmax.

•**Training**: Use `'Categorical Crossentropy'`, optimize with Adam or SGD, and apply data augmentation.
•**Evaluation**: Test accuracy and analyze performance metrics.

Advantages:
•Automatically learns features.
•Effective at handling image data.
•Scalable for larger datasets.

CNN Model Architecture for Fashion MNIST

Model Overview
Input: 28×28×1 grayscale images
Architecture Type: Sequential CNN
Output: 10 fashion categories

Key Features
- Progressive feature extraction
- Increasing filter complexity (32→64→128)
- Two pooling layers for dimension reduction
- ReLU activation for non-linearity
- Softmax for multi-class classification

Model Characteristics
- Lightweight architecture
- Standard CNN pattern
- Suitable for simple image classification
- Efficient parameter usage

Layer Configuration
Convolutional Blocks

Layer	Filters	Kernel Size	Output Shape
Conv2D_1	32	3×3	26×26×32
MaxPool_1	-	2×2	13×13×32
Conv2D_2	64	3×3	11×11×64
MaxPool_2	-	2×2	5×5×64
Conv2D_3	128	3×3	3×3×128

Dense LayersFlatten → 1,152 neurons
Dense_1 → 128 neurons (ReLU)
Output → 10 neurons (Softmax)

CNN Model Performance Analysis

Training and Validation Metrics

Accuracy Trends

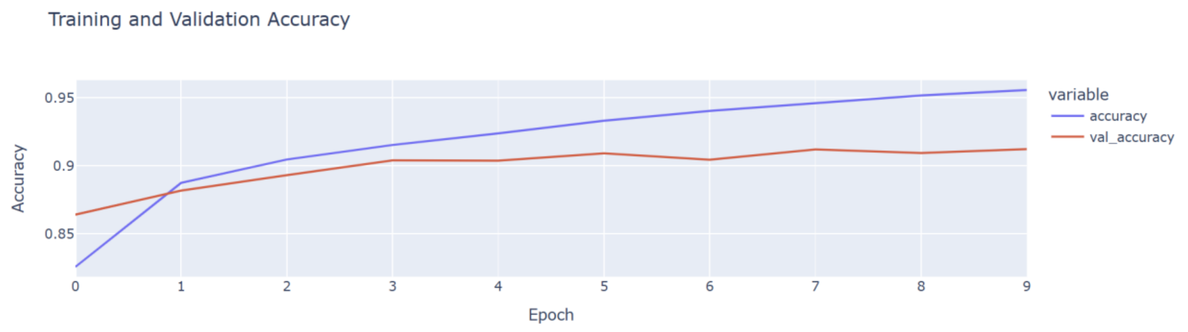
- Training accuracy reaches ~96% after 18 epochs.
- Validation accuracy stabilizes around 91%.
- Gap between training and validation (~5%) suggests slight overfitting.

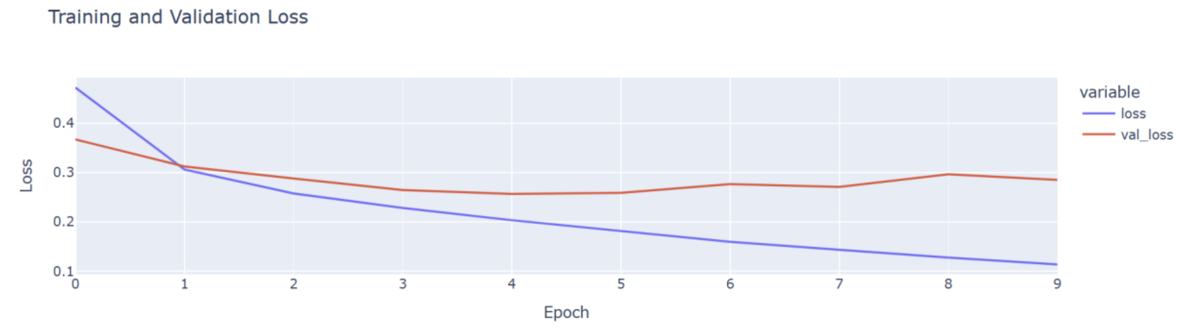
Loss Patterns

- Training loss decreases steadily from 0.45 to 0.12.
- Validation loss stabilizes around 0.28.
- Loss curves indicate model convergence.

Key Observations

Metric	Training	Validation
Final Accuracy	96%	91%
Final Loss	0.12	0.28
Convergence	Continuous improvement	Stable with fluctuations





CNN Model Performance Analysis

Model Characteristics

- Rapid Initial Learning: Sharp improvement in first 2 epochs.
- Consistent Performance: Steady gains after epoch 6.
- Generalization: Good balance between training and validation metrics.
- Stability: Validation metrics show minor fluctuations.

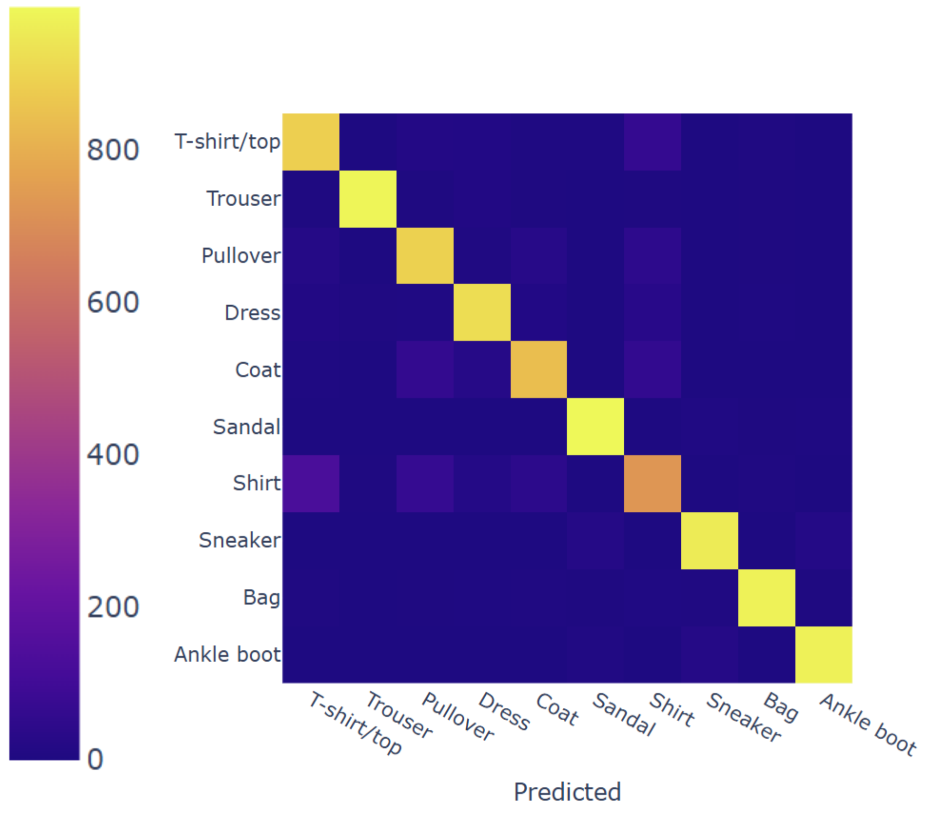
Confusion Matrix Insights

- Overall accuracy: 91%
- Best performers: Trouser (99%), Bag (99%), Sandal (99%)
- Challenging class: Shirt (71% recall, often confused with T-shirt/top)

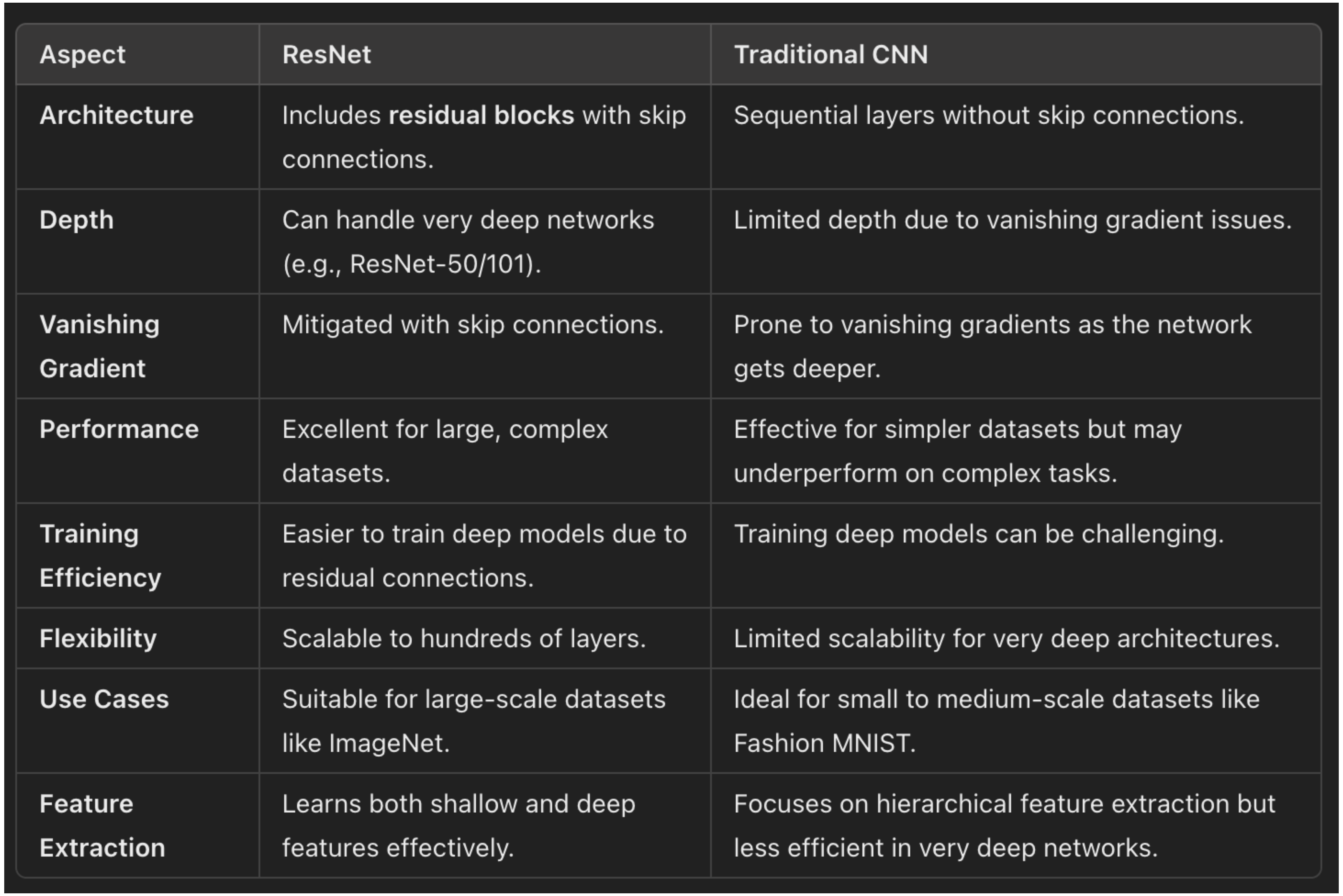
Notable confusions

- Shirt vs T-shirt/top (156 misclassifications)
- Pullover vs Coat (mutual confusion)
- Ankle boot vs Sneaker (some overlap)

This analysis demonstrates robust CNN performance on the Fashion MNIST dataset with specific areas identified for potential improvement.



Difference between ResNet and Traditional CNN



Aspect	ResNet	Traditional CNN
Architecture	Includes **residual blocks** with skip connections.	Sequential layers without skip connections.
Depth	Can handle very deep networks (e.g., ResNet-50/101).	Limited depth due to vanishing gradient issues.
Vanishing Gradient	Mitigated with skip connections.	Prone to vanishing gradients as the network gets deeper.
Performance	Excellent for large, complex datasets.	Effective for simpler datasets but may underperform on complex tasks.
Training Efficiency	Easier to train deep models due to residual connections.	Training deep models can be challenging.
Flexibility	Scalable to hundreds of layers.	Limited scalability for very deep architectures.
Use Cases	Suitable for large-scale datasets like ImageNet.	Ideal for small to medium-scale datasets like Fashion MNIST.
Feature Extraction	Learns both shallow and deep features effectively.	Focuses on hierarchical feature extraction but less efficient in very deep networks.

Other Algorithms

Logistic Regression

- **Purpose**: A simple linear model for classification.
- **Usage**: Flattens 2D Fashion MNIST images to 1D and applies a linear decision boundary.
- **Limitation**: Struggles with complex patterns; best for baseline comparison.

Support Vector Machine (SVM)

- **Purpose**: Finds the optimal hyperplane to separate classes.
- **Usage**: Flattens images; kernel methods (e.g., RBF) help capture non-linear patterns.
- **Limitation**: Computationally expensive for large datasets like Fashion MNIST.

Random Forest

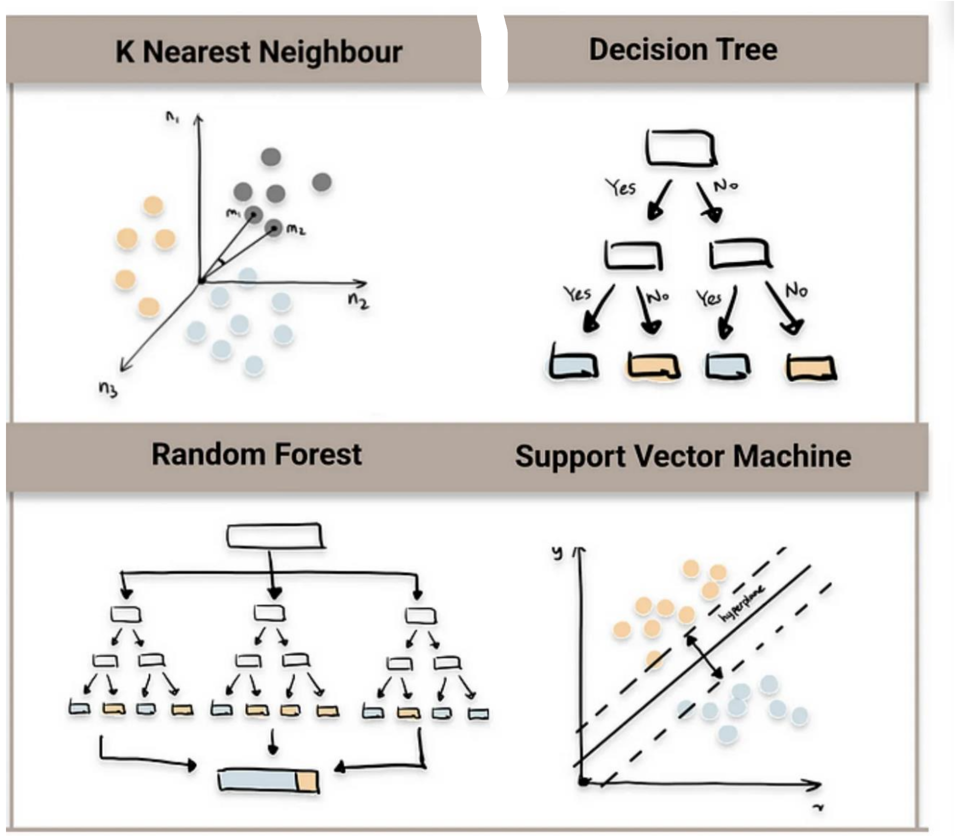
- **Purpose**: Ensemble learning using decision trees.
- **Usage**: Treats flattened features as input, aggregates tree outputs for classification.
- **Limitation**: Less effective than deep learning models for high-dimensional image data.

K-Nearest Neighbors (KNN)

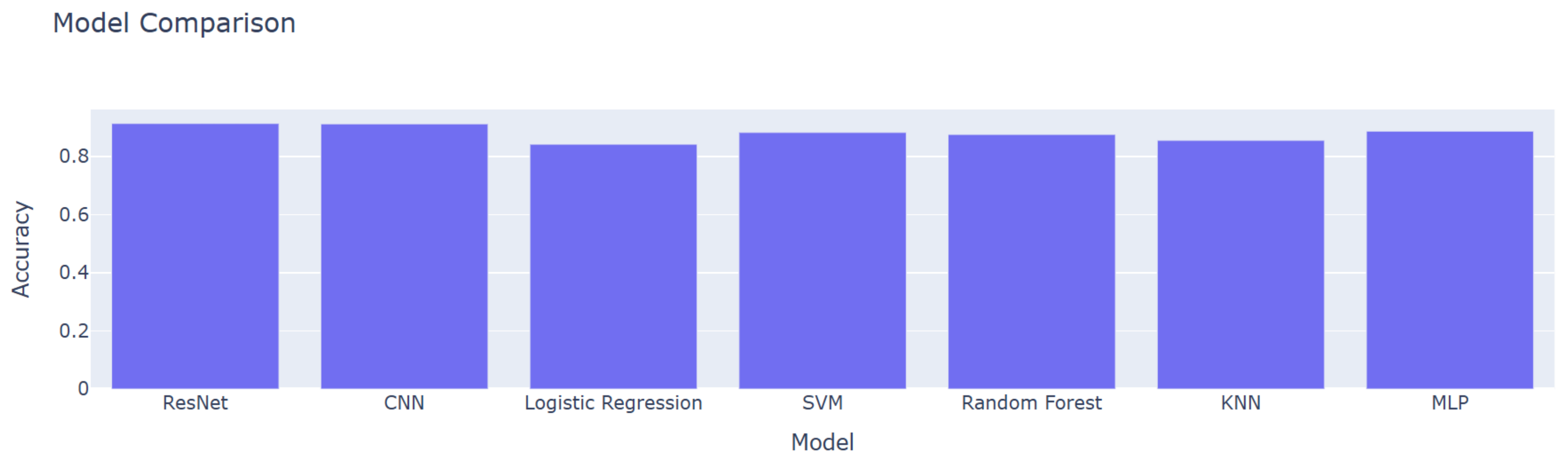
- **Purpose**: Classifies based on the closest labeled samples.
- **Usage**: Works on flattened data, uses a distance metric (e.g., Euclidean).
- **Limitation**: High memory usage and slower for large datasets.

Multi-Layer Perceptron (MLP)

- **Purpose**: Fully connected neural network for non-linear patterns.
- **Usage**: Requires flattened images; captures complex relationships through hidden layers.
- **Limitation**: Less effective than CNNs for image data without spatial awareness.



Traditional ML vs Deep Learning Comparison



Key Insights

- Deep learning models (ResNet, CNN) slightly outperform traditional ML
- Logistic Regression achieves ~83% accuracy despite simplicity
- SVM and Random Forest show competitive performance (~87-89%)
- Traditional methods require flattening 28×28 images to 784 features
- Simpler models offer faster training and better interpretability

This comparison demonstrates that while deep learning achieves superior results, traditional ML methods remain viable for this classification task with reasonable accuracy-complexity trade-offs

Why Compare?

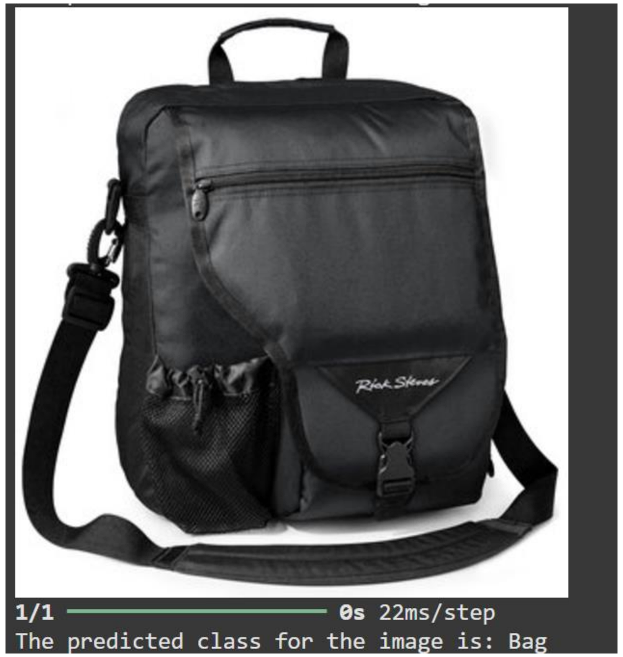
- Validates the need for complex deep learning models.
- Establishes performance benchmarks.
- Helps understand trade-offs between complexity and accuracy.

Model Performance Overview

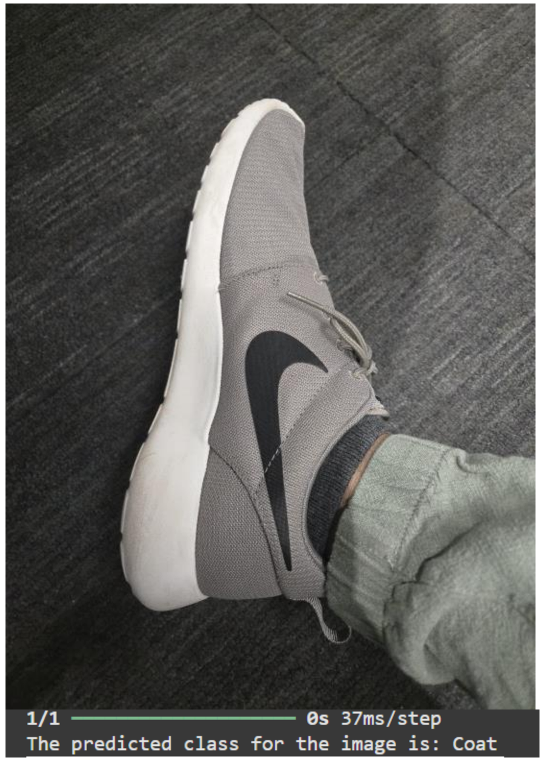
Model Type	Accuracy	Complexity
ResNet/CNN	~91-92%	High
Traditional ML	83-89%	Lower

Arizona State University

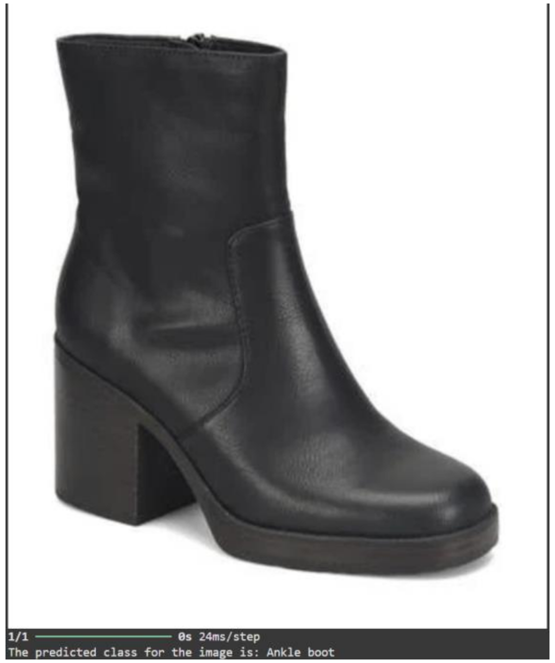
Dataset issues



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
The predicted class for the image is: Bag
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 37ms/step
The predicted class for the image is: Coat
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
The predicted class for the image is: Ankle boot
```

- Dataset bias: Grayscale images with white backgrounds limit real-world applicability
- Model struggles with colored images and complex backgrounds, misclassifying common items