MACHINE LEARNING

TERM PROJECT

Image Object Recognition

COMP 462

Group 6

Fatma Zehra Bayir

Berat Kaya

Yağmur Parmaksız

Kerim Ercan

TABLE OF CONTENTS


01
ABOUT THE PROJECT

Project Description

- **Objective:** This project solves the image object recognition task with CIFAR-10 dataset and create machine learning models.

  Models Used: Multi-Layer Perceptron, Convolutional Neural Network

- **Goal:** Our primary goal is to construct, train, and evaluate machine learning models capable of classifying CIFAR-10 images effectively.

- **Tools we used:**

Google colab

Google Docs

Google Slides

02 DATASET DESCRIPTION

CIFAR-10 Dataset

Feature	Description
Total Images	60,000 (32x32 RGB)
Number of Classes	10
Train/Test Split	50,000 / 10,000
Image Format	32 × 32 × 3 (RGB)
Labels	Integers from 0 to 9

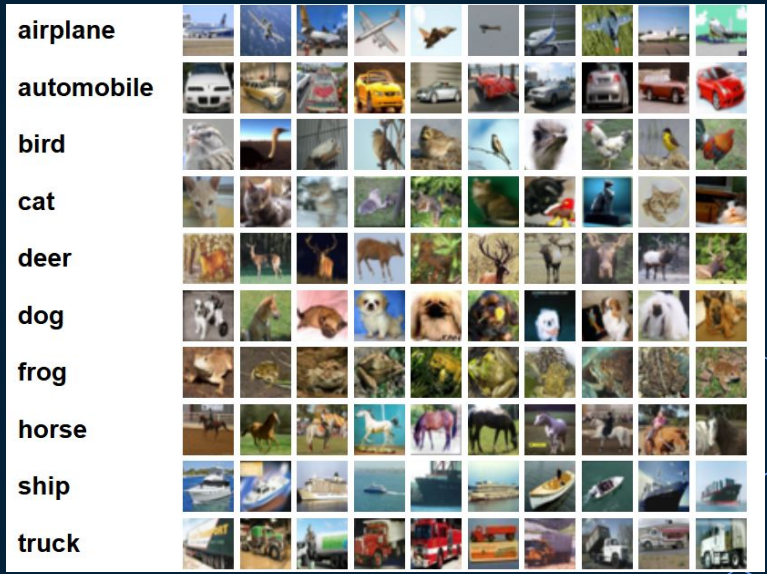


Figure 1. CIFAR-10 Dataset(CIFAR-10 and CIFAR-100 Datasets, 2025)

03 DATA PREPROCESSING

General Steps

1. **Normalized pixel values from 0–255 to 0–1**

2. **Labels:**

   One-hot encoded for neural networks
   Integer labels (0–9) for classical models

3. **Dataset split:**

   50,000 training images
   10,000 test images

Why Are These Steps Important?

Normalization helps models train faster and prevents large gradient updates.

Label encoding ensures compatibility with loss functions like categorical cross entropy.

Consistent train/test split provides a fair evaluation and avoids data leakage.

CNN Preprocessing

- Images kept in original 32×32×3 format
- Pixel normalization (0–1)
- One-hot label encoding
- Dataset split:
  - 80% training, 20% validation
- **Data Augmentation:**
  - Random rotation
  - Horizontal shift
  - Horizontal flip

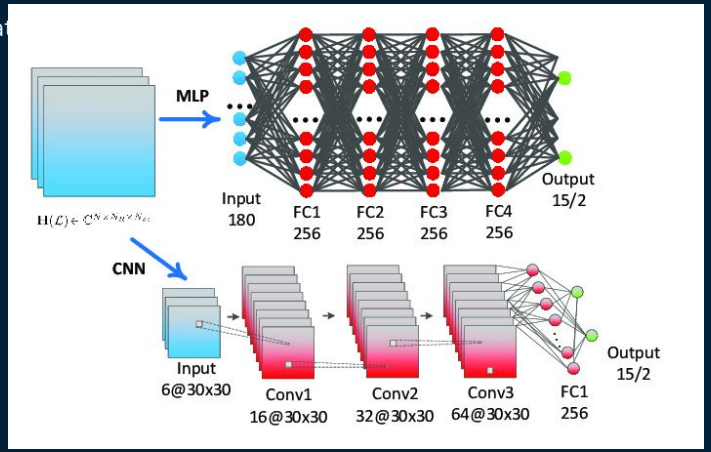


Figure 2.

MLP Preprocessing

- Images flattened to 1D vectors (32×32×3 → 3072)
- No resizing or cropping applied
- Pixel normalization (0–1)
- Labels one-hot encoded
- No data augmentation used

04 FEATURE ENGINEERING

Raw Pixels as Features

- Raw pixel values were directly used as features.
- CIFAR-10 images: **32×32 pixels**, **3 RGB channels**.
- This approach is common and effective for low-resolution datasets.

No Manual Feature Extraction

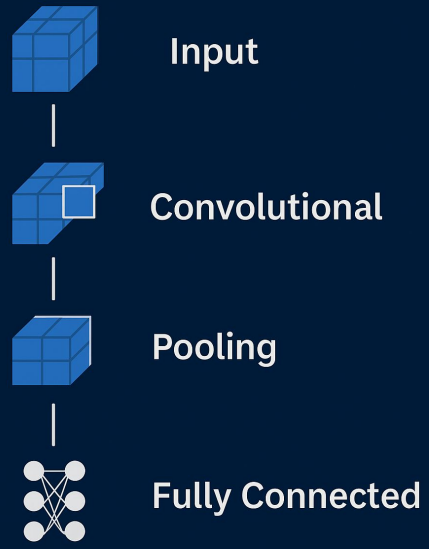
- No handcrafted features, no filters, no classical image processing.
- Models learned features **directly from data**.

MLP Models

- Images were **flattened** into 1D vectors (32×32×3 = 3072).
- Fully connected layers require fixed-length inputs.
- The model tried to learn features from flattened data.

CNN Model

- Images **kept in original 32×32×3 format**.
- Convolutional layers preserved spatial relationships.
- The model learned **local patterns** like edges and textures.



CNN Model Architecture

Result & Performance

- CNN achieved **85% test accuracy** — significantly better than MLPs.
- This proves CNN's strength in **learning spatial features**.
- Deep learning models performed well **without manual feature design**.

05

MULTILAYER PERCEPTRON MODEL

Multilayer Perceptron Model for CIFAR-10

A multi-layer perceptron (MLP) is a type of artificial neural network consisting of multiple layers of neurons. The neurons in the MLP typically use nonlinear activation functions, allowing the network to learn complex patterns in data.[1]

In the project, it was preferred as a flat classification model that can be directly applied to visual data.

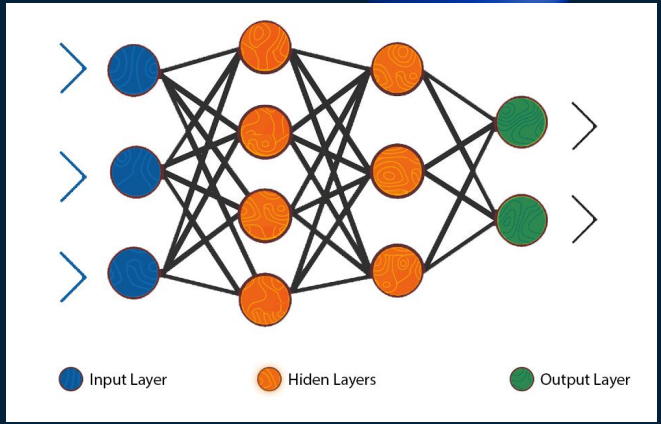


Figure 3. Multilayer perceptron model.

Input layer → Hidden layers → Output layer

Why We Choose MLP

We wanted to use it as a basic reference model to classify low-resolution images in CIFAR-10 data.

MLP offers a simpler architecture compared to CNN and is suitable for quick testing of model performance.

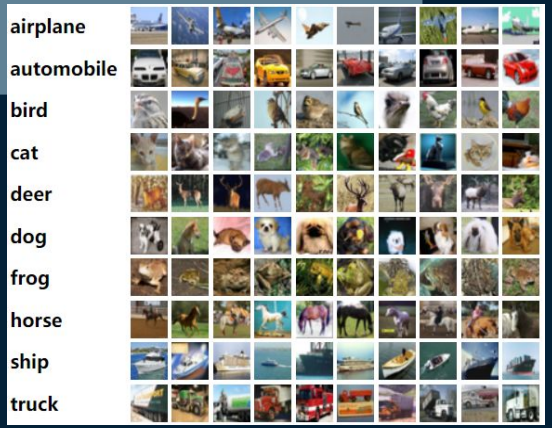


Figure 4. CIFAR-10 Dataset(CIFAR-10 and CIFAR-100 Datasets, 2025)

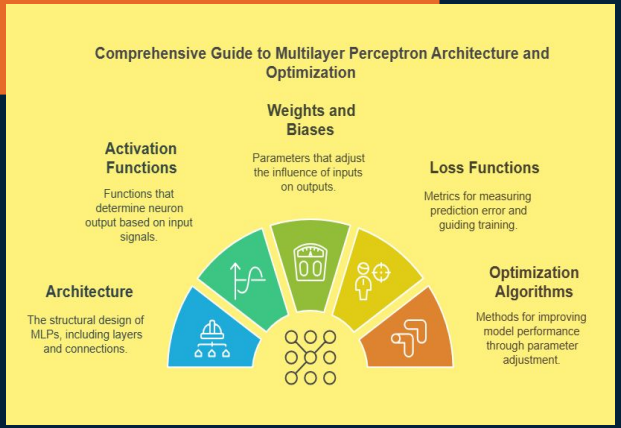


Figure 5. Guide to MLP Architecture

Structure and Implementation of MLP Model

Input Size

Each image of size 32 × 32 × 3 is flattened to a vector of 3072 features.
(32 × 32 × 3 = 3072)

Hidden Layers

Dense Layer 1: 1024 neurons + BatchNorm + ReLU + Dropout(0.3)

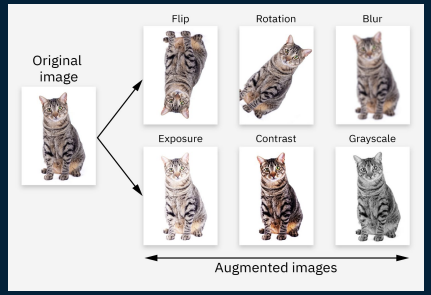
Dense Layer 2: 512 neurons + BatchNorm + ReLU + Dropout(0.3)

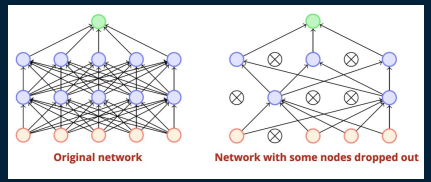
Dense Layer 3: 256 neurons + BatchNorm + ReLU

Output Layer

Dense(10) + ***Softmax***
(corresponds to 10 CIFAR-10 classes)
Softmax activation was used for 10 classes in the output.

Structure and Implementation of MLP Model





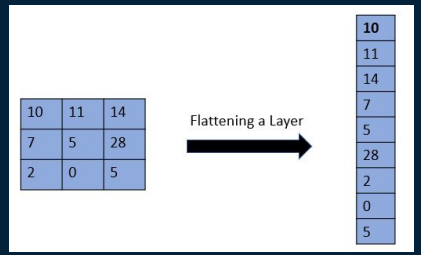


Figure 6,7,8. MLP Structure.

Data Augmentation

CIFAR-10 data was normalized and data augmentation was applied.

Batch Normalization & Dropout

Over-learning was prevented with 3 hidden layers (1024-512-256 neurons), batch normalization and dropout in the model.

Flattening

The input layer flattened the image and converted it to a vector (Flatten).

3072 (Flattened Input) → Dense(1024) → Dense(512) → Dense(256) → Dense(10)

Activation Function

Softmax activation was used for 10 classes in the output.

📈Model Performance Summary on Test Data

Accuracy	58.37%	The percentage of examples that the model correctly classified
Precision	58.71%	The rate of examples that the model calls “correct” being actually correct. It shows that false positives are low.
Recall	58.37%	It shows how many of the true positive examples it correctly predicted. The success of recognizing classes.
F1 Score	57.71%	The balanced average of precision and recall. It gives an idea about the overall consistency of the model.

📊Epoch vs. Accuracy

In the first epochs	Accuracy started around 35%.
Between 3rd and 10th epochs	Accuracy gradually increased to 50% levels.
After 10th epoch	ReduceLROnPlateau was activated and the learning rate was reduced → accuracy started to increase again.
Last epoch (30)	Accuracy: 58.37%, which means the model can generalize.

Epoch means that the model passes through the entire training data once. During this process, the model processes the images, calculates its errors, and updates its weights according to these errors. This model was trained for 30 epochs.

Result of MLP Model

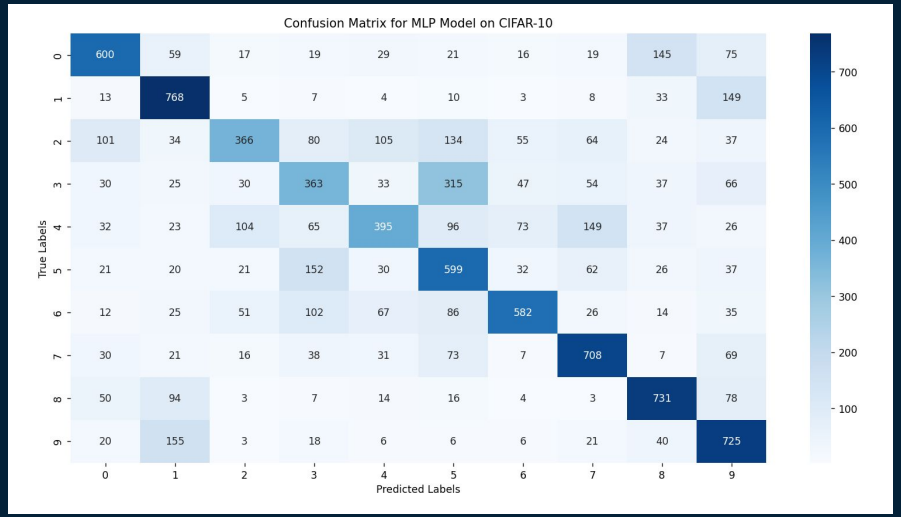


Figure 9. Confusion matrix for MLP.

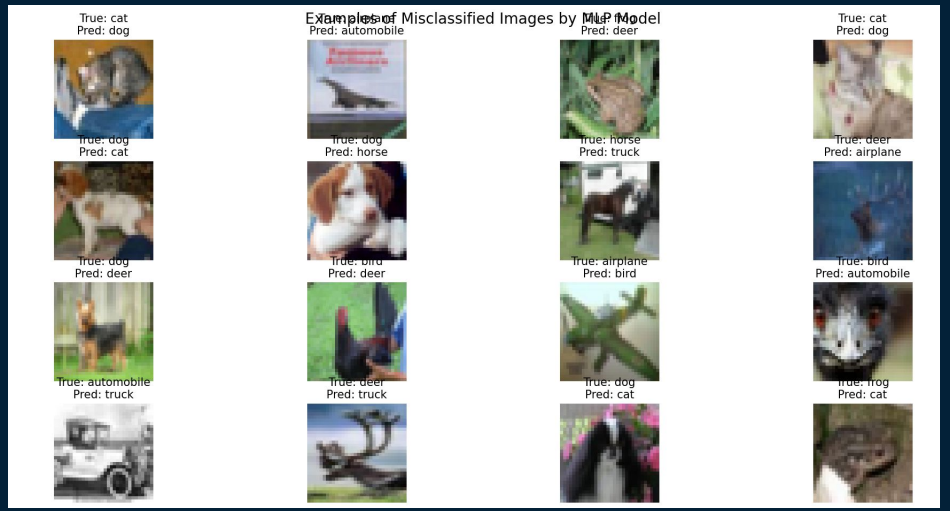


Figure 10. Misclassified images by MLP model.

06
CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Network (CNN) for CIFAR-10

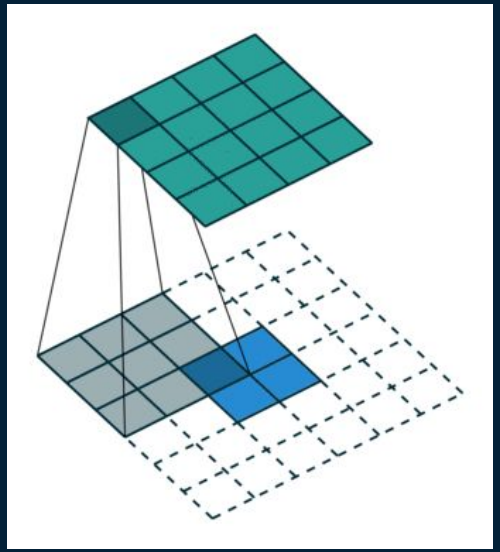


Figure 11.Getting Spatial Patterns

Input layer → Convolutional Layers → Hidden layers →Pooling→ Fully Connected Layers → Output layer

Why We Choose CNN

"We wanted to capture spatial patterns in images like edges and textures, which MLPs cannot detect well."

CNNs are more effective and scalable for image classification tasks and performed significantly better than MLP in our project.

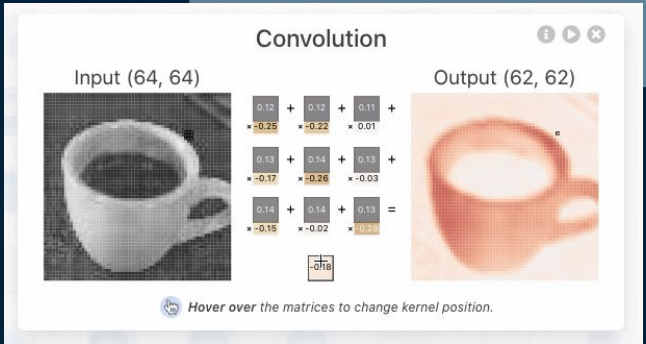


Figure 12.Spatial Patterns

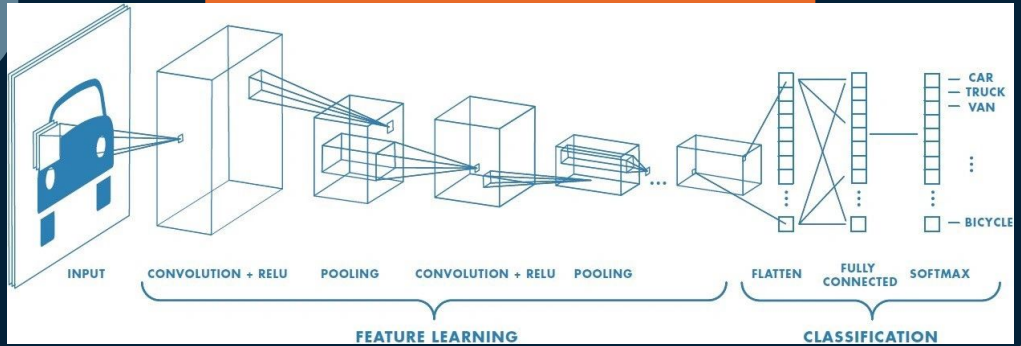
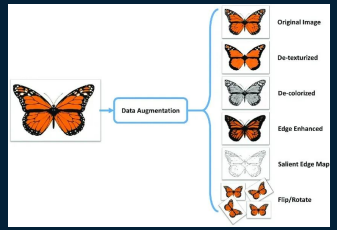


Figure 13.Basic CNN Model

Structure and Implementation of MLP Model

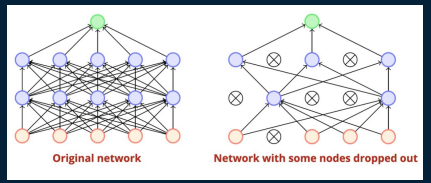
Input Size	Each image of size 32 × 32 × 3 is flattened to a vector of 3072 features. ***(32 × 32 × 3 = 3072)***
Convolutional Blocks	Block 1: Conv2D(32) → BatchNorm → Conv2D(32) → BatchNorm → MaxPooling → Dropout(0.2) Block 2: Conv2D(64) → BatchNorm → Conv2D(64) → BatchNorm → MaxPooling → Dropout(0.3) Block 3: Conv2D(128) → BatchNorm → Conv2D(128) → BatchNorm → MaxPooling → Dropout(0.4)
Output Layers	GlobalAveragePooling2D → Dense(256) → BatchNorm → Dropout(0.5) Dense(10) + Softmax

Structure and Implementation of CNN Model



Data Augmentation

Random rotation/Horizontal shift/Horizontal flip

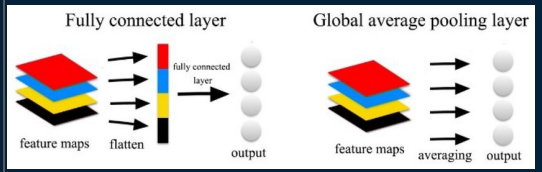


Batch Normalization & Dropout

Each convolutional block used BatchNorm for faster convergence
and Dropout (0.2 / 0.3 / 0.4) to reduce overfitting risk.

Preserved Image Structure

Unlike MLP, images were not flattened.The 2D spatial structure of the image (32×32×3) was preserved throughout the convolutional layers.



Activation Function

ReLU was used after each convolutional layer.
The final layer used Softmax to output probabilities for the 10 CIFAR-10 classes.

Figure 14,15,16.Data Augmentation,Dropout,Fully Connected Layers vs Global Average Pooling Layer .

📈Model Performance Summary on Test Data

Accuracy	86%	The percentage of examples that the model correctly classified
Precision	0.861%	The rate of examples that the model calls “correct” being actually correct. It shows that false positives are low.
Recall	0.859%	It shows how many of the true positive examples it correctly predicted. The success of recognizing classes.
F1 Score	0.857%	The balanced average of precision and recall. It gives an idea about the overall consistency of the model.

Confusion Matrix

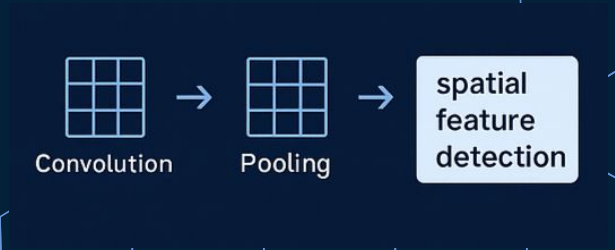
Figure 17. Confusion matrix of CNN.

07

MODEL EVALUATION AND COMPARISON

Model	Accuracy	Precision (avg)	Recall (avg)	F1-Score (avg)
MLP	0.584	0.587	0.584	0.577
CNN	0.850	0.854	0.850	0.848

- **CNN significantly outperformed the MLP model** in all metrics, with more than **26% higher accuracy**.
- The **CNN's ability to learn spatial features** (like edges, textures, and patterns) gave it a clear advantage in understanding image structure.
- In contrast, **MLP treats each pixel independently**, flattening the input and losing valuable spatial relationships.
- MLP struggled particularly with fine-grained classes like **cats, birds, and deer**, where spatial awareness is crucial.
- CNN's use of **convolutional and pooling layers** allowed it to extract meaningful local patterns, resulting in stronger generalization.



08

CHALLENGES

Figure 18. Challenges and learnings about project.

References

[1]Jaiswal, S. (2024, February 7). Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. Datacamp.com; DataCamp. https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

[Figure 1]CIFAR-10 and CIFAR-100 datasets. (2025). Toronto.edu. https://www.cs.toronto.edu/~kriz/cifar.html [Figure 3]AIML.com. (2022, June 27). What is a Multilayer Perceptron (MLP) or a Feedforward Neural Network (FNN)? AIML.com. https://aiml.com/what-is-a-multilayer-perceptron-mlp/

[Figure 5]Multiayer Perceptrons (MLP) in Machine Learning Explained. (2024). Code B Website https://code-b.dev/blog/guide-to-multilayer-perceptrons-in-machine-learning

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... & Zheng, X. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems (arXiv:1603.04467). https://arxiv.org/abs/1603.04467

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Chollet, F. (2015). Keras (Version 2.x) [Computer software]. https://keras.io

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. Foundations and Trends® in Signal Processing, 7(3–4), 197–387. https://doi.org/10.1561/2000000039

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern

Jaiswal, S. (2024, February 7). Multilayer Perceptrons in Machine Learning: A Comprehensive Guide. Datacamp.com; DataCamp. https://www.datacamp.com/tutorial/multilayer-perceptrons-in-machine-learning

Recognition (CVPR) (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, 25, 1097–1105. https://doi.org/10.1145/3065386

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML) (pp. 807–814).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929–1958.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In European Conference on Computer Vision (ECCV) (pp. 818–833). Springer. https://doi.org/10.1007/978-3-319-10590-1_53