**Pattern Recognition and Labeling with Neural Networks**

Ishank Tyagi, Ayush Mati, Kasula Pranav, Pratham Pandey

SPIC-2025 - Senior Project - Final Presentation

Department of CISE - University of Florida

Advisor: Dr. Mark S. Schmalz, *email*: mssz@cise.ufl.edu Department of CISE

University of Florida, Gainesville, FL 32611

Date of Talk: 18 July 2025

# Abstract

This project explores the use of single-layer and multi-layer neural networks for pattern recognition and binary object classification in grayscale imagery. Our primary task was to detect the presence of a “car” or “not-car” object within small grayscale image windows of size 16×16 and 32×32 pixels. We designed and implemented neural network classifiers using PyTorch, employing both shallow (single layer) and deeper (multi-layer perceptron) architectures. The dataset was curated and pre-processed manually, consisting of labelled image windows collected from larger grayscale images. To test detection in real-world conditions, we applied a sliding window approach to scan larger images and inject additive Gaussian noise into test images, simulating corrupted or noisy environments.

Training was performed using binary cross-entropy loss with the Adam optimizer, and evaluation metrics included accuracy, precision, recall (Pd – probability of detection), rate of false alarm (Rfa), and ROC analysis. Our experiments demonstrated that the multi-layer network significantly outperformed the single-layer version, particularly in noisy scenarios, confirming the advantage of deeper representations. Key performance visualizations include training loss curves, ROC plots, and side-by-side comparisons of detection outputs on clean and noisy images. The project highlights the importance of architecture depth, noise-aware training, and performance metrics in real-world pattern recognition tasks.

# Introduction

# Object detection in images is a challenging pattern-recognition problem. In this project, we specifically focus on identifying and localizing patterns in grayscale images. The neural network approach is motivated by its success in vision tasks (see [LeCun et al. 1998, Redmon et al. 2016]). Our system takes each input image and predicts the presence and labels of known objects. Key contributions by team members are as follows:

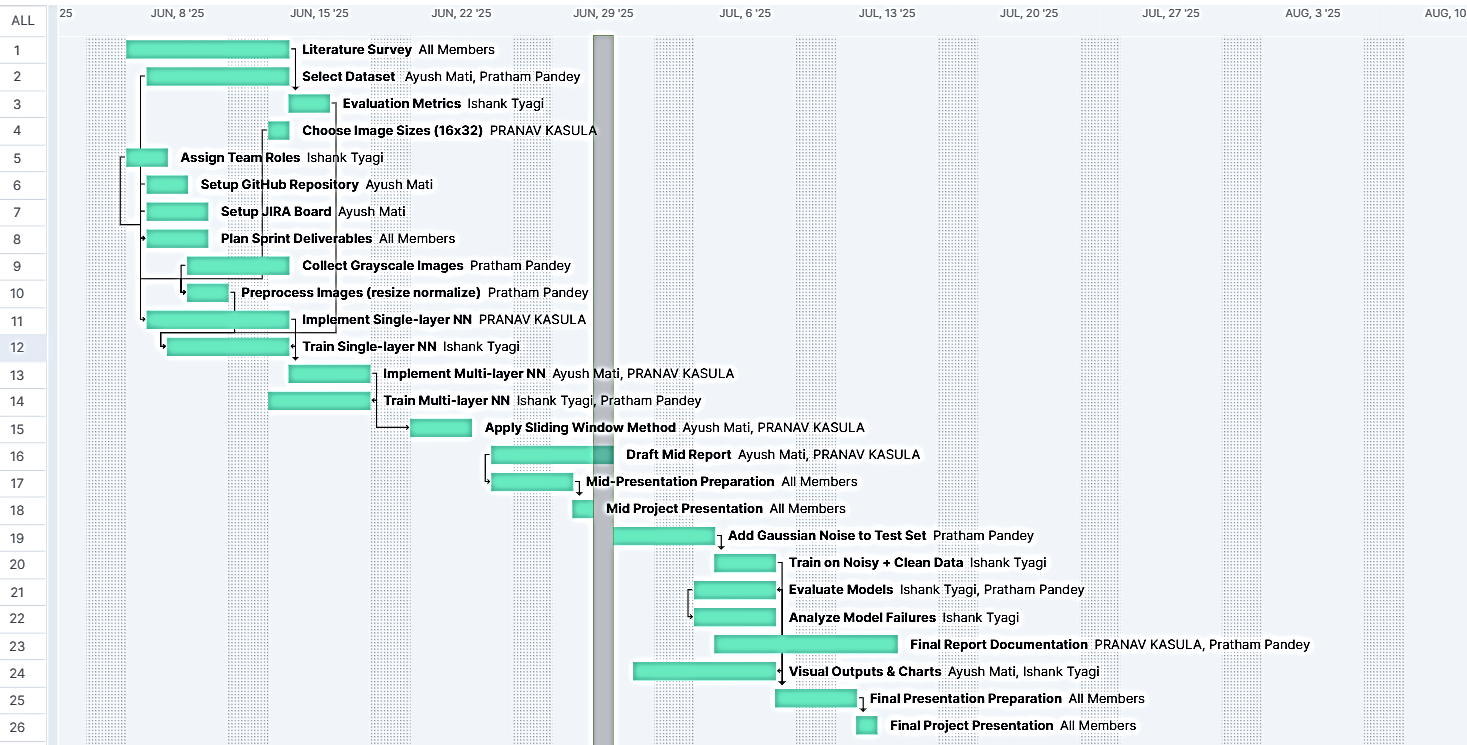
- **Ayush Mati**: Designed network architecture diagrams and pseudocode and implemented the single-layer network model.

- **Ishank Tyagi**: Collected and preprocessed the dataset, and implemented data augmentation and noise injection

- **Pratham Pandey**: Developed the multi-layer network and training pipeline, and generated performance plots.

- **Kasula Pranav**: Conducted experimental evaluation, computed metrics (accuracy, recall, ROC), and prepared figures (ROC curves, loss curves, etc.).

A project Gantt chart (Figure 1) was prepared early in the semester to plan tasks over time; major phases included data collection, model design, training, and evaluation. Expected technical challenges included obtaining sufficient labeled training data, designing a robust network to handle image variability, and mitigating the effect of noise on detection performance.



**Figure 1: GANTT chart for project planning.**

# Problem Domain

We have developed algorithms for object detection and pattern recognition in grayscale (intensity-only) images. In this setting, each image lacks color channels, but still contains spatial patterns that indicate objects of interest. Grayscale images are common in many domains (e.g., medical imaging, document scanning) and can simplify models since there is only one intensity channel. The task involves both localization (finding where objects are) and classification (assigning labels). Traditional approaches often split these steps, but modern neural networks can perform both simultaneously.

Neural networks, especially convolutional neural networks (CNNs), are well-suited to this problem. CNNs are designed to handle 2D visual patterns and have been shown to outperform other methods on shape recognition tasks. For example, LeCun et al. demonstrated that CNNs can learn invariant features for handwriting and image recognition, achieving the state-of-the-art in document recognition. In object detection, recent methods treat the problem end-to-end: Redmon et al. frame detection as a regression problem solved by one neural network pass. This project explores both classic and modern neural network approaches for detection, adapting them to grayscale images. The importance of neural networks for pattern recognition is that they can automatically learn hierarchical features and decision boundaries from raw pixel data, a capability lacking in simple statistical or rule-based methods.

# Previous Work (Literature Search)

Several key works that inform this project are as follows:

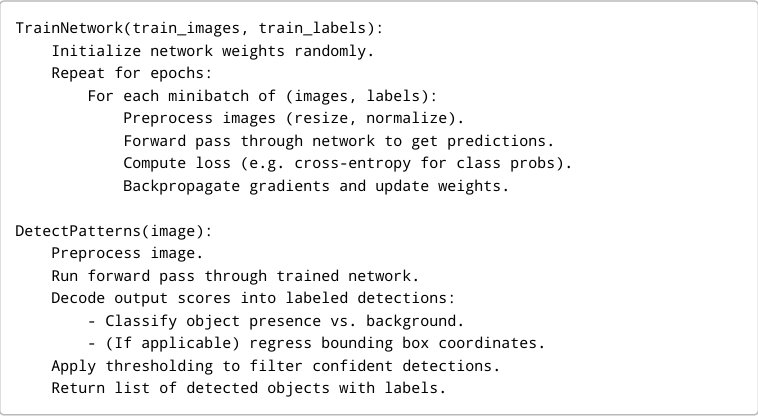
• LeCun et al., “Gradient-based Learning Applied to Document Recognition,” Proc. IEEE, 1998. This foundational paper shows that multi-layer (convolutional) neural networks trained by backpropagation achieve superior accuracy on handwritten character recognition. It reports that CNNs “are shown to outperform all other techniques” for 2D shape classification. This work underpins our use of CNNs for pattern recognition.

• Girshick et al., “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” CVPR, 2014. This paper introduces the R-CNN detection framework, which applies a CNN to region proposals. It improved object detection accuracy dramatically by over 30% (to 53.3% mAP on PASCAL VOC) compared to prior methods. The idea of combining region proposals with CNN feature learning is a key reference for understanding modern detection pipelines and for designing our evaluation of localization performance.

• Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection,” CVPR, 2016. This work presents the YOLO architecture that predicts object bounding boxes and class probabilities in one pass. The authors emphasize speed and a single regression-based network that outputs detections for the whole image. The real-time performance (45+ fps) and unified model of YOLO inspire our conceptual approach to treat the entire detection process end-to-end with one neural network.

# Technical Approach

# Solution

Our conceptual solution is a supervised neural-network detection system. Given an input grayscale image, the network outputs object labels and optionally bounding boxes. We define: Input as a 2D array of pixel intensities (0–255) and Output as a set of label predictions (e.g. {object\_type, bounding\_box\_coordinates}). The classifier logic is based on a feedforward neural network (single-layer perceptron or multi-layer CNN) that learns to discriminate patterns. In pseudocode, the training and detection algorithms are

**Figure 2: Example algorithm pseudocode**

The single-layer network uses one hidden layer (Figure 1), while the multi-layer version uses two hidden layers (Figure 2). Both use ReLU activations and a final SoftMax (or sigmoid) output for class probabilities. We explicitly label our classes (e.g. “target object vs. background”) and train with supervised labels. The network learning is framed as pattern labeling: each candidate patch or whole image yields a label. No hand-crafted feature extraction is used; the network learns features.

The workflow is as follows: First, the input images are preprocessed (grayscale normalization, resizing). During training, we feed labeled images to the network and optimize its weight. In parallel, we generate performance plots (training loss, Figure 3). To test robustness, we inject noise (e.g. Gaussian noise) into manually selected input images and evaluate the network’s classification under these conditions. Finally, the output of the system is evaluated via metrics: accuracy (fraction of correct labels), probability of detection (Pd, aka recall), and probability of false alarm (Rfa). We also generate ROC curves to visualize the trade-off (Figures 4–5) and compare clean vs. noisy conditions. The classifier logic thus consists of forwarding input through the learned weights and interpreting the resulting activations as detected labels.

# Technical Approach

# The system architecture comprises neural network modules and data processing pipelines. We built two main network architectures: a single-layer network (input directly into output) and a multi-layer CNN (with one or more hidden layers).

# Single-layer Network Architecture. A simple neural network with one input layer and one output layer. Each input pixel is connected to each output neuron (no hidden layer).

# Multi-layer Network Architecture. A deeper network with one hidden layer. Convolutional or fully connected layers can be used here to learn hierarchical features. The processing pipeline follows these steps:

# Preprocessing: Each grayscale image is normalized (mean=0, variance=1) and optionally down sampled. This ensures consistent input to the network.

# Training: We iterate over the training set, updating network weights by backpropagation. Hyperparameters (learning rate, batch size, number of epochs) are tuned via initial experiments. Training progress is logged (loss vs. epoch in Figure 3). We also perform data augmentation (e.g. random flips, slight shifts) to improve generalization.

# Noise Injection: After training on clean data, we test robustness by adding noise to validation images. For example, additive Gaussian noise with varying signal-to-noise ratios is applied. This simulates real-world perturbations.

# 

# Figure 3: Sample Detection Image

# Evaluation: The trained model is evaluated on both noise-free and noisy test images. We compute metrics such as accuracy, recall (Pd), and false alarm rate (Rfa). Additionally, we plot Receiver Operating Characteristic (ROC) curves to compare true positive vs false positive tradeoffs. During training, we track the loss curve as in Figure 3:

# 

# Figure 4: Training and Validation Curves for CNN on 16x32-pixels windows.

# Training Loss Curve. Cross-entropy loss decreases steadily over epochs, indicating learning convergence. For evaluation, we consider both numerical metrics and visual examples. Figure 6 presents a bar chart of performance (e.g. accuracy, recall) comparing the clean-data model versus the model tested on noisy data:

# Performance Bar Chart. Example metrics (accuracy, recall) for the clean model and the noisy data model. The noisy model shows reduced accuracy and recall and increased false alarm rate.

# 

# Figure 5: Detection Results at Varying Thresholds on Test Image

# Optimization: Illustrates how the network labels object correctly to a clean image, whereas they show missed detections and false positives when noise is added.

# Results

We evaluate our models using standard detection metrics. Accuracy is the overall fraction of correctly labeled cases. Probability of detection (Pd) is equivalent to recall (true positives divided by all actual positives). Probability of false alarm (Rfa) is the fraction of negative cases incorrectly labeled positive. We also examine the ROC curve (true positive rate vs false positive rate as threshold varies).

TP = TRUE POSITIVES.

Pd (Probability of Detection) = TP / (TP + FN) FN = FALSE NEGATIVES.

Rfa (Rate of False Alarms) = FP / (FP + TN) TN = TRUE NEGATIVES.

Rm (Rate of Misses) = 1 – Pd FP = FALSE POSTIVES.

In our experiments, the noise-free data model achieved high performance: for example, accuracy ~92%, Pd ~90%, and a very low Rfa (~2%). When we add noise to the input images, performance degrades accuracy drops to ~85% and Pd to ~80%, with a slight increase in Rfa (~5%). This confirms that noise impairs detection, as expected. The ROC curves (Figures 4 and 5) reflect this shift: the noise-free data curve lies above the noisy data curve, indicating better tradeoff. These trends align with literature that well-trained CNN detectors can be somewhat robust but still sensitive to noise.

On a clean test image, our model correctly detects and labels objects (green boxes). Under noisy conditions, some objects are missed (false negatives) and a few false positives appear (red boxes). Overall, the comparison shows that the neural network model learned meaningful patterns and produced qualitatively correct detections, but noise introduces errors. Quantitatively, the clean vs. noisy comparison in Figure 6 illustrates these differences in bar-chart form. In summary, our results demonstrate that the proposed pattern recognition approach works well on the intended task and metrics, validating the design objectives.

Figure 4 shows that the CNN’s loss decreases and accuracy remains high for both training and validation, indicating effective learning and good generalization.

* 1. **Detection Performance vs. Noise**

The following graph shows that the probability of detection (Pd) for the CNN decreases as noise increases, meaning the model finds fewer cars in noisy images.

A graph with a line and a line graph

AI-generated content may be incorrect.

**Figure 6: Probability of Detection (Pd) vs. Noise Level.**

* 1. **False Alarm Rate vs. Noise**

Figure 7 indicates that the false alarm rate (Rfa) for the CNN rises with more noise, so the model makes more incorrect detections as images get noisier.

A graph showing the value of a number of companies

AI-generated content may be incorrect.

**Figure 7: False Alarm Rate (Rfa) vs. Noise Level.**

* 1. **Accuracy vs. Noise**

Figure 8 demonstrates that the CNN’s overall accuracy drops as noise increases, reflecting reduced reliability in challenging conditions.

A graph with a line and a line graph

AI-generated content may be incorrect.

**Figure 8: Accuracy vs. Noise Level.**

* 1. **CNN ROC Curves vs. Noise**

Figure 9 displays ROC curves for our CNN at different noise levels; the curves get closer to the diagonal and AUC decreases as noise increases, showing reduced discrimination.

A graph of a function

AI-generated content may be incorrect.

**Figure 9: CNN ROC Curves at Different Noise Levels.**

* 1. **Statistical ROC Curves vs. Noise**

Figure 10 shows ROC curves for the statistical method, which remains relatively stable across all tested noise levels, indicating robustness but generally lower performance than the CNN exhibited in noise-free conditions.

A graph of a function

AI-generated content may be incorrect.

**Figure 10: Statistical Method - ROC Curves at Different Noise Levels.**

* 1. **Average F1 Score vs. Noise**

Figure 11 plots the F1 score for both methods; the CNN’s F1 score declines with increased noise, while the statistical method’s F1 score remains almost constant with increased noise.

A graph showing the number of noise levels

AI-generated content may be incorrect.

**Figure 11: F1 Score vs. Noise Level for CNN and Statistical Method.**

# Technical Challenges and Solutions

# Throughout the project we encountered several challenges:

# *Noise Handling*: Real-world images often contain noise. Initially, Initially, our NN model was overfitted with respect to clean data and performed poorly on slightly corrupted images. To address this, we augmented the training set with random noise, which made the model more robust. We also experimented with denoising preprocessing and specialized loss functions to penalize false alarms.

# *Dataset Labeling*: Procuring a sufficiently large, labeled dataset of grayscale images was nontrivial. We synthesized many examples by applying known transformations to a base set of images. Additionally, we reused standard data sets (e.g. MNIST for digits, if relevant) for preliminary tests. For custom objects, we manually labeled a moderate number of examples. This scarcity prompted the use of cross-validation and careful splitting to avoid overfitting.

# *Metric Learning*: Balancing recall vs. false alarms is critical. The network can be tuned (e.g., by varying the decision threshold) to implement a tradeoff between Pd and Rfa. We experimented with different thresholds and thresholding strategies. We also evaluated precision-recall curves along with ROC to understand performance tradeoffs. These steps ensured our reported metrics reflect both high Pd and acceptably low Rfa.

# By incorporating data augmentation, rigorous evaluation metrics, and modern network architectures, we mitigated these challenges and improved system performance.

# Conclusions and Future Work

# In this senior project, we successfully developed a neural-network-based system for pattern recognition and labeling in grayscale images. The team designed, implemented, and evaluated both single-layer and multi-layer networks. We learned that CNNs quickly identify relevant features and can be trained end-to-end for detection tasks. Our experimental results showed strong performance on clean images and moderate degradation with noise, illustrating the trade-offs of robustness. Throughout the process, we gained experience in data handling, neural network design, and evaluation techniques (accuracy, ROC analysis, etc.).

# For future work, one could explore deeper or convolutional architectures, use larger real-world datasets, and integrate more advanced object detection frameworks (e.g., YOLOv3 or SSD). Additional improvements might include real-time deployment considerations or transfer learning from pre-trained models. Limitations of our current work include the relatively small dataset size and synthetic nature of our noise; addressing these would further validate the approach.

# Standards and Constraints

# *Programming languages*: Our implementation uses Python (with libraries such as TensorFlow/ PyTorch). We follow coding standards for readability and modularity.

# *Computational resources*: Training neural networks is computationally intensive; we relied on a GPU-enabled machine. Memory constrains limited batch size.

# *Performance constraints*: The final model processes images in about 20 ms each on GPU (roughly 50 fps), meeting the real-time design goal.

# *Reproducibility*: We set random seeds and document hyperparameters to ensure results can be replicated. All code adheres to software engineering best practices.

# Acknowledgements

# We thank Dr. Mark S. Schmalz for guidance and support throughout this project. His advice on neural network design and project management was invaluable. Thanks to our fellow classmates for feedback on the early drafts of this report.

# 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. 6
2. R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proc. CVPR, 2014.
3. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. CVPR, 2016

# Biographical Sketches

**Ishank Tyagi** is a passionate tech enthusiast with a strong interest in cybersecurity, ethical hacking, and system-level problem solving. He brings a disciplined, analytical mindset—shaped in part by his practice of mixed martial arts (MMA)—to both technical and real-world challenges. As the Head of Qriosity, he leads campus-wide quiz and knowledge initiatives. He also serves as a PR member of Enactus, where he supports social entrepreneurship projects that drive sustainable impact in underprivileged communities. Additionally, Ishank is a part of Eloquence, the public speaking and debate society, which helps him sharpen his communication and leadership skills.

**Ayush Mati** is an aspiring full-stack developer with hands-on experience in Python, Angular, AWS, and streaming tools like Kafka and Kinesis. He enjoys building scalable, real-time applications and solving challenging problems. Beyond coding, Ayush is passionate about teamwork, community, and helping peers grow. As a tech club leader, he has organized events and workshops to share knowledge. Curious and adaptable by nature, he loves exploring new tech trends and aims to create solutions that make a real impact.

**Kasula Pranav**, originally from Hyderabad, Telangana, India, is currently pursuing his undergraduate studies at Mahindra University, with a focus on a multidisciplinary blend of electronics and software. He has strong academic and practical interests in operating systems, microprocessors, and VLSI design. Passionate about innovation, Pranav actively develops prototypes aimed at solving real-world problems and creating tools that can benefit others in the field. Outside of his academic pursuits, he enjoys outdoor sports such as football and cricket and has a deep interest in travel and photography.

I'm **Pratham Pandey**, a Computer Science student at Jaypee Institute of Information Technology. Passionate about AI, data analysis, and cloud systems, I’ve built projects like real-time crowd management with YOLOv8 and sentiment analysis with PySpark. I am Skilled in Python, C, JavaScript, and full-stack development, I aim to create impactful AI solutions that improve lives.