Project Report: Multi-Model MNIST Classifier

(a) System Structure and Tools I Used

- My goal was to guess the correct number (0 to 9) from the pictures.
- I used a **Stacking Generalization Ensemble** at the end to get the best score.

Base Tools (Layer 0)

- **Softmax Tool (Discriminative):** This was my best starting tool. It uses **Softmax Loss** and **Gradient Descent** to find the best line to separate the numbers. It gives a **Probability** score for each number.
- **OVR Linear Tool (Discriminative):** I used this for a simple straight-line guess for each number (One-vs-Rest).
- **Gaussian Tool (Generative):** This finds the average shape and size of each number using **density estimation**. It gives a **Log Score** showing how well the picture matches the average shape.
- **KMeans Tool (Clustering):** This puts the pictures into the closest of 12 groups I made. It gives a simple **1 or 0** guess.

Stacking Final Plan (Layer 1)

- **New Data:** I took the scores from the four basic tools. This gave me **40 new features** for every picture (4×10).
- **Final Tool:** I trained a **second Softmax Tool** on these 40 new features.
- This final tool learned the **best way** to mix all the scores from the four basic tools to get the final, best answer.

(b) Settings I Tested and Final Scores

- I tested many settings (hyper-parameters) on a small set of data to make each tool work perfectly.

Individual Tool Performance and Settings

- **Softmax Tool:**
  - Best settings: **L_2 control (λ)=1e-5**, **speed (α)=1e-2**.
  - Final score: **0.9062 Macro F1**.
- **OVR Linear Tool:**
  - Best settings: **L_2 control (λ)=1e-5**, **speed (α)=1e-2**.

    - Final score: **0.8473 Macro F1**.
- **Gaussian Tool:**
    - Best setting: **Smoothing (ϵ)=1e-4**.
    - Final score: **0.7086 Macro F1**.
- **KMeans Tool:**
    - Best setting: **$k=12$ groups**.
    - Final score: **0.5810 Macro F1**.

Final Ensemble Performance

- **Simple Weighted Mix:** My best simple mix (Softmax:5, OVR:1, Gaussian:1, KMeans:0) got **0.9125 Macro F1**.
- **Stacking Plan (Best):** My Stacking Plan got the highest final score: **$\mathbf{0.9160}$ Macro F1** and **$\mathbf{0.9163}$ Accuracy**.

(c) How I Made It Fast and Accurate

Getting the Highest Score (Accuracy)

- **Scaled the Pictures:** I made the pixel colors normal ($X/255.0$). This was critical for making **Gradient Descent** run stable and fast.
- **Used Stacking:** Using the **Stacking Plan** gave me the final best score, increasing the result by almost $\mathbf{0.01}$ F1 over the best single tool.

Making It Fast (Under 5 Minutes)

- **Total Time:** All training took about **50 seconds**, which is very fast (well under the 5-minute limit).
- **Fast Math:** I used **NumPy** for all the heavy math.
- **Controlled Steps:** I set the "maximum steps" for the slower tools (Softmax, OVR) to **500**. This was enough for the math to reach the best answer without wasting extra time.

(d) What I Saw and Learned

- **Softmax is Strongest:** The **Softmax Tool** was the best basic tool. This shows that the numbers are **linearly separable** (easy to separate with a straight line) in the data space.
- **Mixing Works Best:** My score improved every time I mixed the tools. This proves that no single tool is perfect.
- **Smart Error Correction:** The **Stacking Plan** worked best because the final tool learned to trust the strong Softmax tool, but used the scores from the weaker tools (like Gaussian) to fix small errors the Softmax tool missed.
- **Efficiency:** I built a powerful system using only basic math rules in Python, proving I can get great results while staying very fast and efficient.