Course: UCS761
Lab 7-Assignment
Piyush Taneja
102303307

1️⃣ Objective

In this experiment, we compared three optimizers:

- SGD
- Momentum
- Adam

The goal was to analyze:

- Convergence speed
- Stability of training
- Final loss behavior

The comparison was performed on the CNN model using the synthetic 8×8 image dataset.

2️⃣ Observations from the Convergence Plot

From the loss vs epoch graph:

🔹 SGD

- Starts with highest loss (~0.19)
- Gradually decreases
- Converges slowly
- Final loss ≈ 0.055 after 5 epochs

Observation:
SGD shows steady but slow convergence. The loss decreases linearly but does not reach very low values within few epochs.

Interpretation:
SGD updates parameters using only the current gradient, which can lead to slower progress especially when gradients are small or noisy.

🔹 Momentum

- Starts lower than SGD (~0.13)
- Converges faster than SGD

- Final loss ≈ 0.033

Observation:
Momentum reduces loss faster than SGD and achieves lower final loss within same number of epochs.

Interpretation:
Momentum accumulates past gradients, which helps accelerate learning in consistent directions and reduces oscillations.

🔹 Adam

- Starts with very low loss (~0.013)
- Reaches near-zero loss by epoch 1
- Remains stable afterward

Observation:
Adam converges extremely fast and achieves near-zero loss almost immediately.

Interpretation:
Adam uses adaptive learning rates for each parameter and combines momentum with RMS scaling. This allows faster and more stable convergence.

3️⃣ Convergence Speed Comparison

Ranking by convergence speed:

1. Adam (Fastest)
2. Momentum
3. SGD (Slowest)

Adam reaches minimal loss within 1–2 epochs.
Momentum improves significantly over vanilla SGD.
SGD requires more epochs for similar performance.

4️⃣ Stability Analysis

- SGD shows smooth but slower decline.
- Momentum is smoother than SGD and less oscillatory.
- Adam converges sharply and remains stable afterward.

No major instability was observed, but SGD shows relatively slower improvement.

---

5 Interpretation in Terms of Optimization Theory

SGD

Updates:

$$w = w - \eta \nabla L$$

- Uses only current gradient.
- Can oscillate.
- Slower convergence in deep models.

Momentum

Updates:

$$v_t = \beta v_{t-1} + \eta \nabla L$$
$$w = w - v_t$$

- Accumulates gradient history.
- Reduces zig-zag behavior.
- Faster descent in consistent directions.

Adam

Updates:

- Uses first moment (momentum)
- Uses second moment (variance scaling)
- Adapts learning rate per parameter

Result:

- Faster convergence
- Better handling of noisy gradients
- More stable updates

---

6 Effect on CNN Training

The CNN model benefits significantly from adaptive optimization.

Because:

- CNN parameters interact spatially.
- Gradients can vary in magnitude.
- Adaptive learning helps stabilize updates.

Adam therefore improves both speed and final performance.

7️⃣ Key Conclusion

- Optimizer choice significantly affects convergence speed.
- Adam achieves fastest and most stable training.
- Momentum improves over SGD.
- SGD is reliable but slower.

In deeper or more complex networks, optimizer choice becomes more critical.

4. Parameter Derivations

4.1 Dense Network – 2 Layer Architecture (2–8–1)

Architecture:

- Input layer = 2 neurons
- Hidden layer = 8 neurons
- Output layer = 1 neuron

For a fully connected layer with
n_{in}inputs and n_{out}outputs:

$$\text{Parameters} = (n_{in} \times n_{out}) + n_{out}$$

Where:

- $n_{in} \times n_{out}$represents weights
- n_{out}represents biases

Layer 1 (Input → Hidden)

Weights:

$$2 \times 8 = 16$$

Biases:

$$8$$

Total parameters:

$$16 + 8 = 24$$

Layer 2 (Hidden → Output)

Weights:

$$8 \times 1 = 8$$

Bias:

$$1$$

Total parameters:

$$8 + 1 = 9$$

Total Parameters (2-layer Network)

$$24 + 9 = 33$$

4.2 Dense Network – 5 Layer Architecture (2–8–8–8–8–1)

Architecture:
2 → 8 → 8 → 8 → 8 → 1

Layer 1 (2 → 8)

$$(2 \times 8) + 8 = 16 + 8 = 24$$

Layer 2 (8 → 8)

$$(8 \times 8) + 8 = 64 + 8 = 72$$

Layer 3 (8 → 8)

$$(8 \times 8) + 8 = 64 + 8 = 72$$

Layer 4 (8 → 8)

$$(8 \times 8) + 8 = 64 + 8 = 72$$

Layer 5 (8 → 1)

$$(8 \times 1) + 1 = 8 + 1 = 9$$

Total Parameters (5-layer Network)

$$24 + 72 + 72 + 72 + 9 = 249$$

4.3 Dense Network – 10 Layer Architecture

(2–8–8–8–8–8–8–8–8–1)

Architecture:
2 → 8 → 8 → 8 → 8 → 8 → 8 → 8 → 8 → 1

Layer 1 (2 → 8)

$$(2 \times 8) + 8 = 24$$

Layers 2 to 8 (each 8 → 8)

Each layer:

$$(8 \times 8) + 8 = 72$$

There are 7 such layers:

$$7 \times 72 = 504$$

Final Layer (8 → 1)

$$(8 \times 1) + 1 = 9$$

Total Parameters (10-layer Network)

$$24 + 504 + 9 = 537$$

4.4 CNN Parameter Derivation

The CNN model consists of:

- One convolution layer
- One dense output layer

Convolution Layer

Formula:

$$(F \times F \times C_{in}) \times C_{out} + C_{out}$$

Where:

- Filter size $F = 3$
- Input channels $C_{in} = 1$
- Output channels $C_{out} = 1$

Weights:

$$3 \times 3 \times 1 = 9$$

Bias:

$$1$$

Total parameters (Conv Layer):

$$9 + 1 = 10$$

Dense Layer After Flatten

After convolution:

Output size = 6 × 6 = 36 neurons

Dense layer:

$$(36 \times 1) + 1$$
$$= 36 + 1$$
$$= 37$$

Total Parameters (CNN)

$$10 + 37$$
$$= 47$$

4.5 Comparison of Dense vs CNN

Dense network (5-layer):
249 parameters

Dense network (10-layer):
537 parameters

CNN:
47 parameters

Observation:

The CNN has significantly fewer parameters due to parameter sharing in convolution. In dense networks, each input is connected to every neuron in the next layer, leading to rapid parameter growth. In contrast, convolution layers reuse the same filter across spatial locations, making the model more parameter efficient and better suited for image dat