27) Model Robustness with Dropout Regularization on MNIST

<u>Name:-</u>

Rishi A Sheth **PES2UG23CS479**

Rithvik Matta **PES2UG23CS485**

<u>Course Name:-</u>

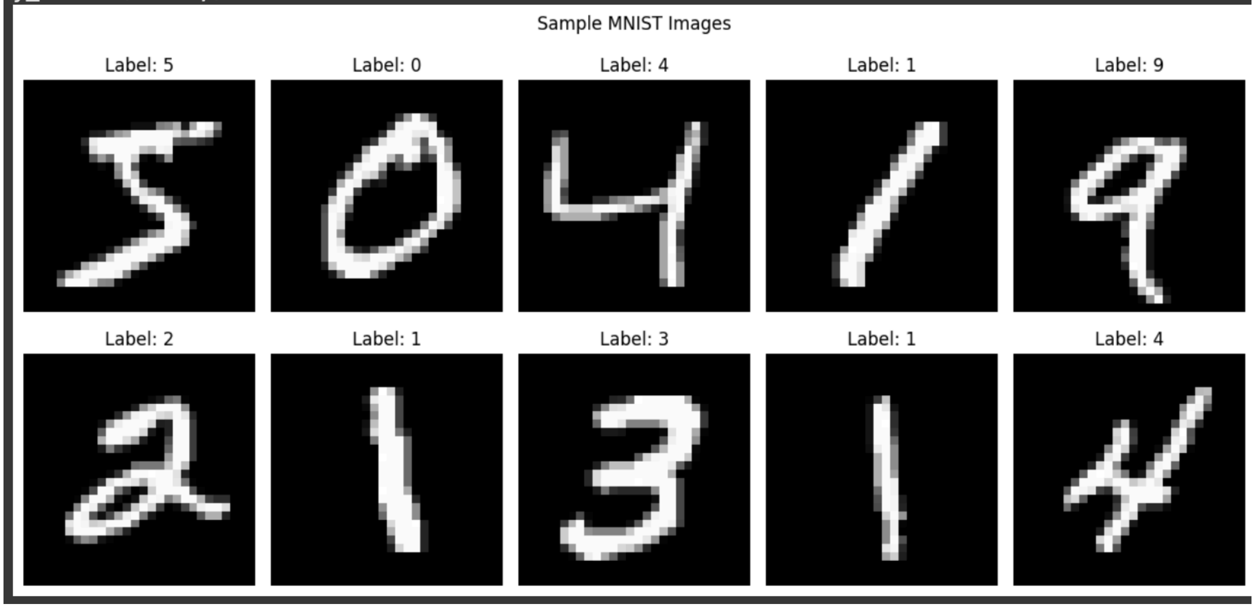
Machine Learning (UE23CS352A)

1.Introduction

1. Deep neural networks can achieve high training accuracy yet generalize poorly. Dropout regularization stochastically masks neurons during training, preventing co-adaptation and improving generalization. This study assesses how varying dropout rates affect accuracy and stability in an MLP trained on MNIST. The goals are to:
    - Compare baseline (0.0) against 0.2 and 0.5 dropout.
    - Quantify generalization (validation accuracy) and stability (variance across runs).

Provide practical guidance for selecting dropout rates in dense MLPs.

```
Original data shapes:
X_train: (60000, 28, 28)
y_train: (60000,)
X_test: (10000, 28, 28)
y_test: (10000,)
```



2. Dataset and Preprocessing

- Dataset: MNIST handwritten digits (60,000 train, 10,000 test; 28×28 grayscale).
- Preprocessing: Flatten to 784-d vectors; normalize to ; one-hot encode labels.
- Split: Standard MNIST train/test split.
- No data augmentation is used to isolate dropout's effect.

3. Methodology

3.1 Architecture

- Input: 784 features.
- Hidden layers: Dense 512 → Dense 256 → Dense 128 with ReLU.
- Regularization: Dropout after each hidden layer at rates {0.0, 0.2, 0.5}.
- Output: Dense(10) with softmax.
- Optimization: Adam; loss: categorical cross-entropy; batch size: 128; epochs: 15.

Rationale: ReLU aids training efficiency; progressive width reduction encourages compression; dropout after each layer regularizes the entire stack.

3.2 Training Protocol

- Environment: Google Colab, TensorFlow 2.x.
- Repetitions: 3 runs per dropout setting for variance estimation.
- Validation: 10,000-image MNIST test split.

- Tracking: Train/val accuracy per epoch; final metrics aggregated as mean ± std.

Model: "sequential"

Layer (type)	Output Shape	Param #
hidden_1 (Dense)	(None, 512)	401,920
dropout_1_0.2 (Dropout)	(None, 512)	0
hidden_2 (Dense)	(None, 256)	131,328
dropout_2_0.2 (Dropout)	(None, 256)	0
hidden_3 (Dense)	(None, 128)	32,896
dropout_3_0.2 (Dropout)	(None, 128)	0
output (Dense)	(None, 10)	1,290

Total params: 567,434 (2.16 MB)
Trainable params: 567,434 (2.16 MB)
Non-trainable params: 0 (0.00 B)

4. Results

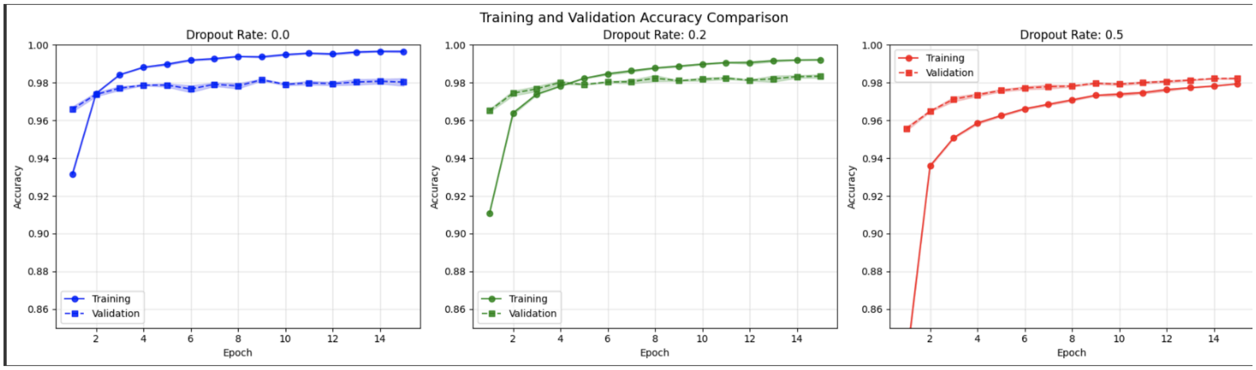
4.1 Quantitative Summary (mean ± std over 3 runs)

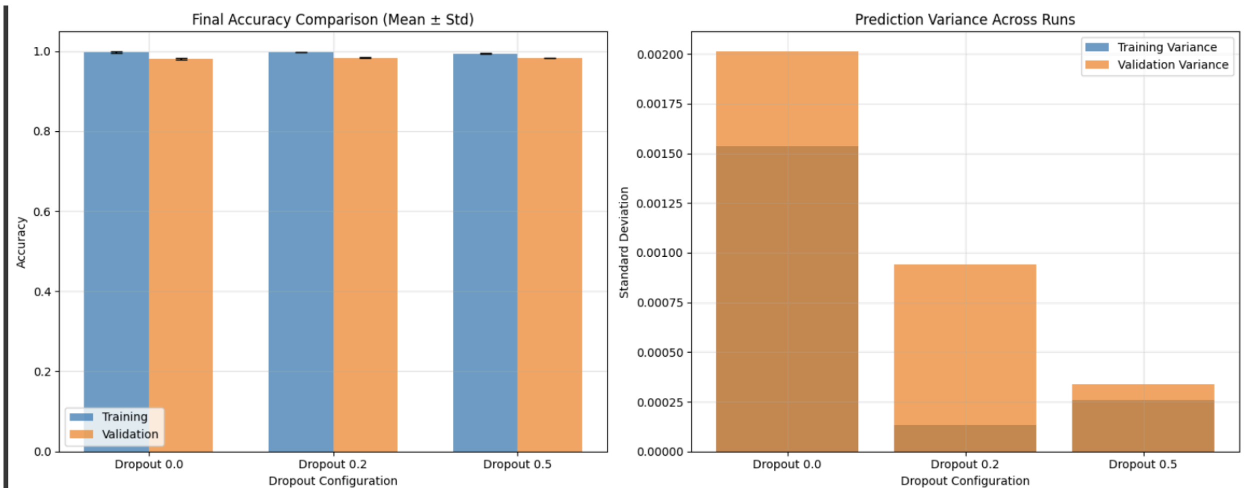
- 0.0 dropout: Train 0.9969 ± 0.0019; Val 0.9803 ± 0.0025; Gap 0.0166.
- 0.2 dropout: Train 0.9978 ± 0.0002; Val 0.9834 ± 0.0012; Gap 0.0144.
- 0.5 dropout: Train 0.9938 ± 0.0003; Val 0.9822 ± 0.0004; Gap 0.0116.

Interpretation: Baseline attains the highest training accuracy but the lowest validation accuracy, indicating overfitting. Dropout 0.2 gives the best validation performance. Dropout 0.5 further lowers the generalization gap and variance, with a small trade-off in mean validation accuracy relative to 0.2.

EXPERIMENTAL RESULTS SUMMARY:
==

Dropout Rate	Train Acc (Mean)	Train Acc (Std)	Val Acc (Mean)	Val Acc (Std)	Overfitting Gap
0.0	0.9969	0.0015	0.9803	0.0020	0.0166
0.2	0.9978	0.0001	0.9834	0.0009	0.0144
0.5	0.9938	0.0003	0.9822	0.0003	0.0116






```
VARIANCE ANALYSIS:
==================================================
Dropout 0.0: Train Variance = 0.000002, Val Variance = 0.000004
Dropout 0.2: Train Variance = 0.000000, Val Variance = 0.000001
Dropout 0.5: Train Variance = 0.000000, Val Variance = 0.000000
```

4.2 Stability and Variance

Validation accuracy standard deviation:

- 0.0: 0.00246
- 0.2: 0.00115 (~53% reduction vs. baseline)
- 0.5: 0.00042 (~83% reduction vs. baseline)

Conclusion: Dropout substantially stabilizes performance across random initializations; 0.5 provides the strongest stabilization.

4.3 Best Configuration and Error Patterns

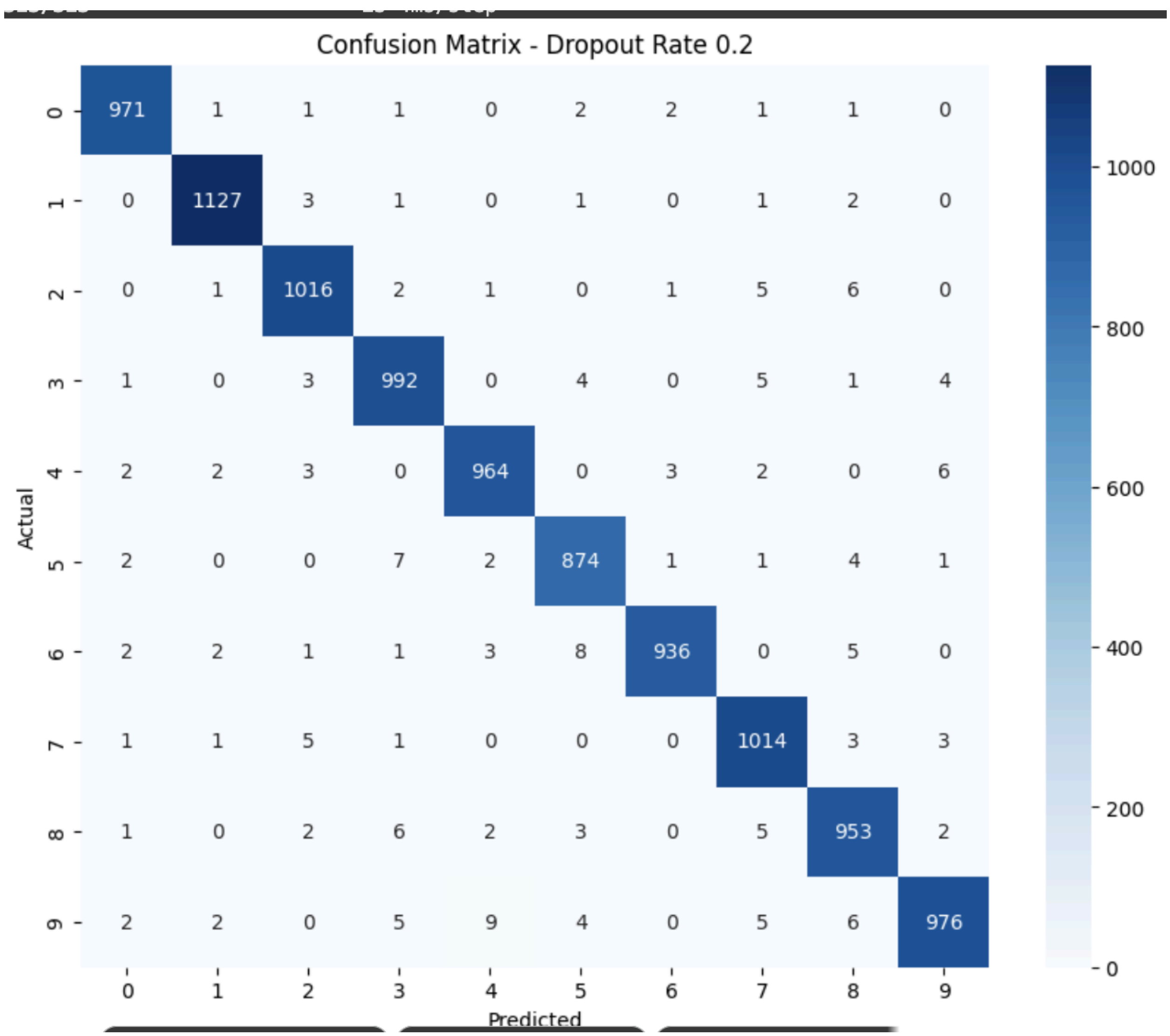
- Best validation accuracy: Dropout 0.2 (0.9834 mean).
- Overfitting gap: Smallest at dropout 0.5 (0.0116), suggesting stronger regularization.

- Typical confusions (qualitative): Visually similar digits (e.g., 4 vs 9, 5 vs 6) tend to be misclassified in MNIST.

5. Discussion

Mechanism of Dropout: Randomly deactivating neurons trains many thinned subnetworks; at inference, using all neurons approximates averaging an ensemble, improving generalization and stability.

Trade-offs: Moderate dropout (0.2) balances capacity and regularization, maximizing validation accuracy. Higher dropout (0.5) slightly reduces accuracy but meaningfully lowers variance and the overfitting gap—useful when consistency across runs is a priority.



Practical guidance:

- Start at 0.2 for dense MLPs on clean datasets like MNIST.
- If training curves show overfitting or results vary across seeds, increase to 0.5.
- Always report mean ± std over multiple runs, and include learning curves and a confusion matrix.

Threats to validity:

- Results use an MLP; CNNs may respond differently to dropout (often applied to fully connected layers).
- Only 3 runs per setting; more repetitions would narrow uncertainty bands.
- Default Adam settings and constant learning rate; scheduler choices may interact with regularization.

6. Conclusion

Dropout improves both generalization and run-to-run stability in an MLP on MNIST. A dropout rate of 0.2 provides the best accuracy, while 0.5 delivers the most stable outcomes with the smallest overfitting gap. Choice of rate should depend on the desired balance between peak accuracy and stability across training runs.

7. Reproducibility and Implementation Notes

- **Environment**: Google Colab; TensorFlow 2.x; Python 3.x.
- **Data**: Keras MNIST loader and standard train/test split.
- **Hyperparameters**: 15 epochs; batch size 128; Adam optimizer.
- **Experimental design**: Dropout $\in \{0.0, 0.2, 0.5\}$; 3 runs per configuration; aggregated metrics as mean ± std.
- **Files**: Attached results file mnist_dropout_results.csv and formatted summary for inclusion in the report.