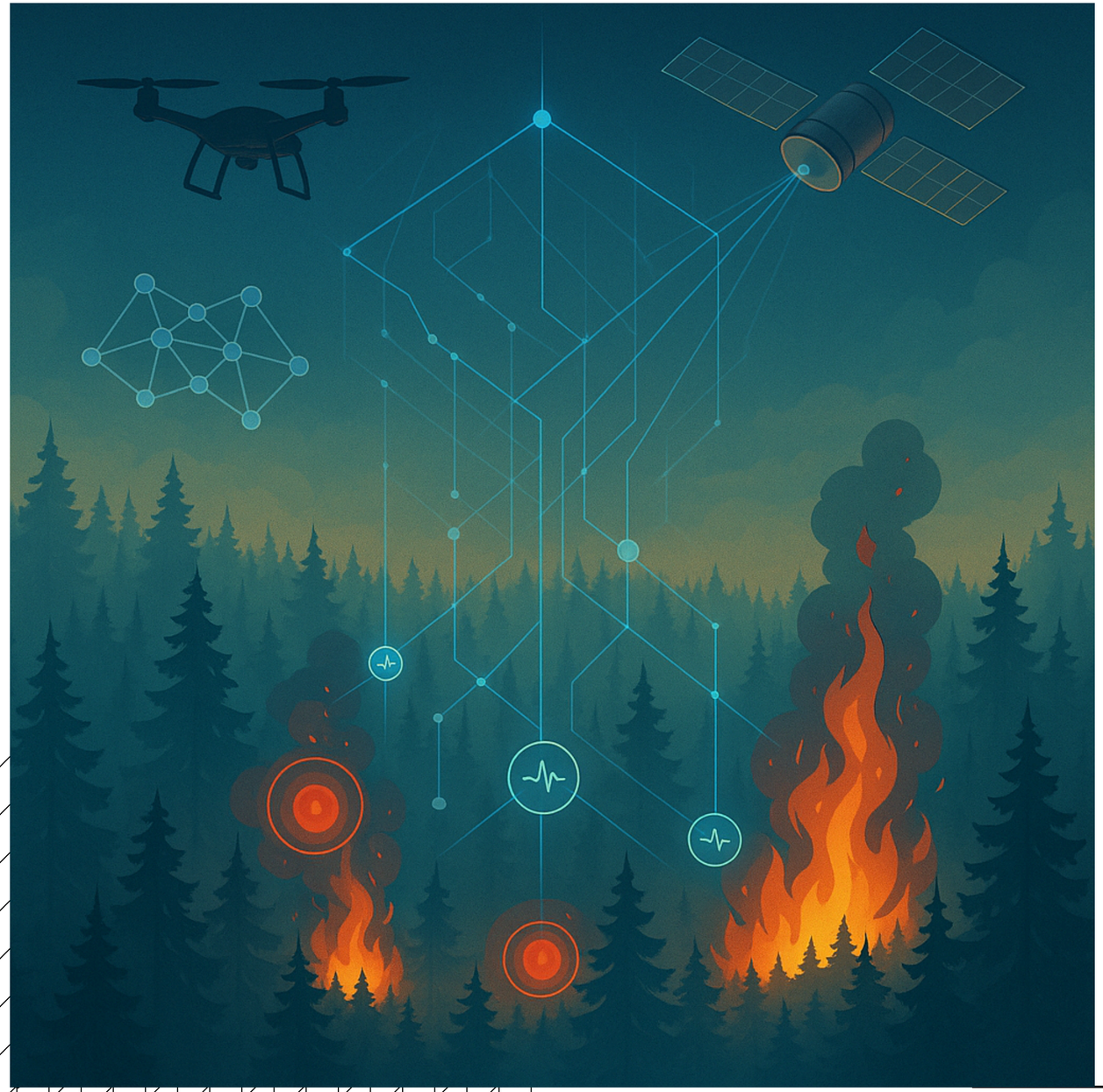


Neural Network-Based Real-Time Forest Fire Early Warning System

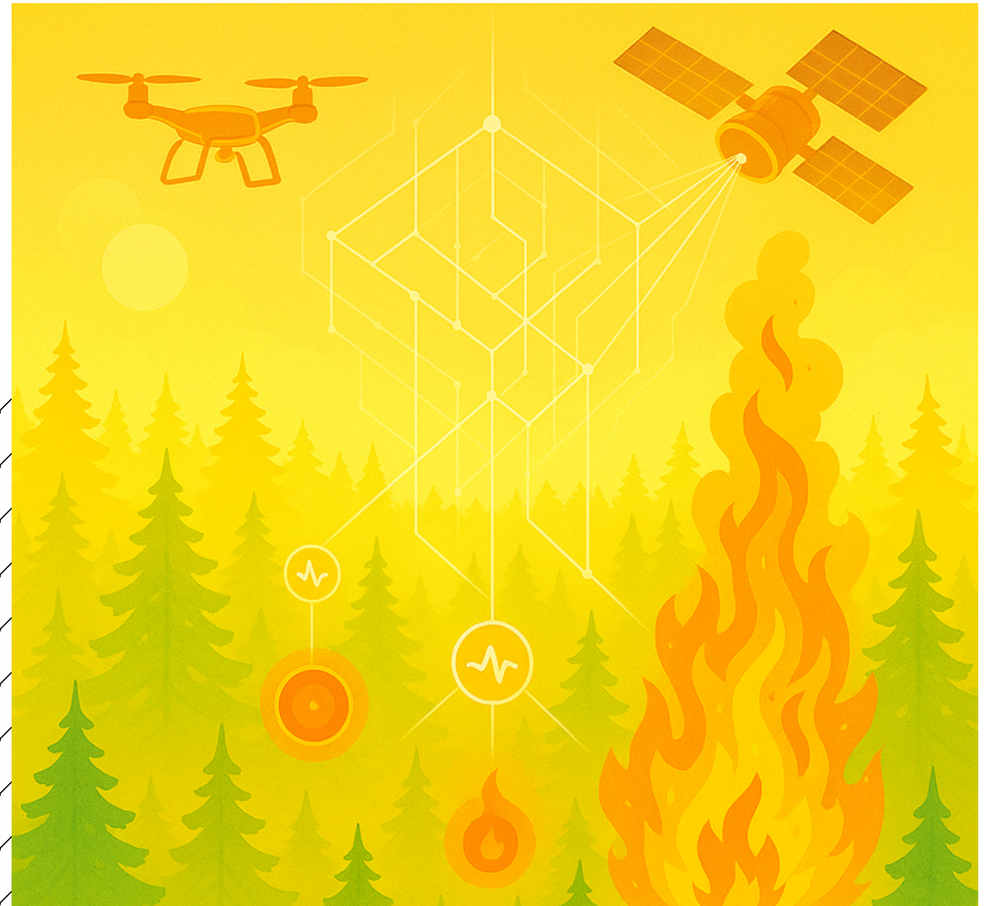
Presentation by

Group A6

Xinru Zhao, Philip Tong, Jie Lin, Kimberly Chen, Yihang Xu, Zihao Li

UC Irvine
Paul Merage
School of Business

Contents of The Report



Introduction

- Outcome variable: Binary (Fire vs. No Fire)
- Business application: Early forest fire detection via image classification.
- Used in surveillance systems (e.g., drones, cameras) for rapid fire response.

Model Evaluation Metrics

- Primary metric: F1-score
- Why F1-score: Balances precision and recall, critical for high-stakes scenarios.
- Recall: Detects actual fire images.
- Precision: Avoids false fire alarms.
- F1-score: Harmonic mean ensuring balance between false negatives and positives.

Human Performance Benchmark

- 100% human classification accuracy on 50-image sample.
- Clear identification of 'No Fire' cases contributed to zero classification error.

Dataset Composition

- Ratio of 'Fire' images: 50% (1,200/2,400)
- Image size: 64x64 RGB (resized during preprocessing)
- All images normalized to same shape: (64, 64, 3)

Dataset Split

- Training: 720 Fire, 720 No-Fire → 1,440 Total
- Validation: 240 Fire, 240 No-Fire → 480 Total
- Test: 240 Fire, 240 No-Fire → 480 Total
- Total: 2,400 images evenly split between classes

Base model

Logistic Regression (No Hidden Layers)

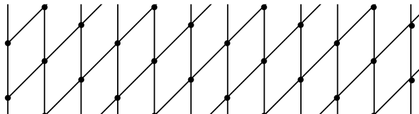
- Train / CV：97.5%
- Test Set：93.54%
- Model: Linear classifier
- Observation: Simple, fast to train, surprisingly effective

Base model

Can Depth Improve Accuracy? Not Always.

- Adding more hidden layers does not always improve performance.
- The best result (95.0%) came from a shallow network with just 2 hidden layers.
- Deeper models (4, 6, 8+) suffered significant drops in accuracy.
- Likely reasons: overfitting, vanishing gradients, and limited training data.

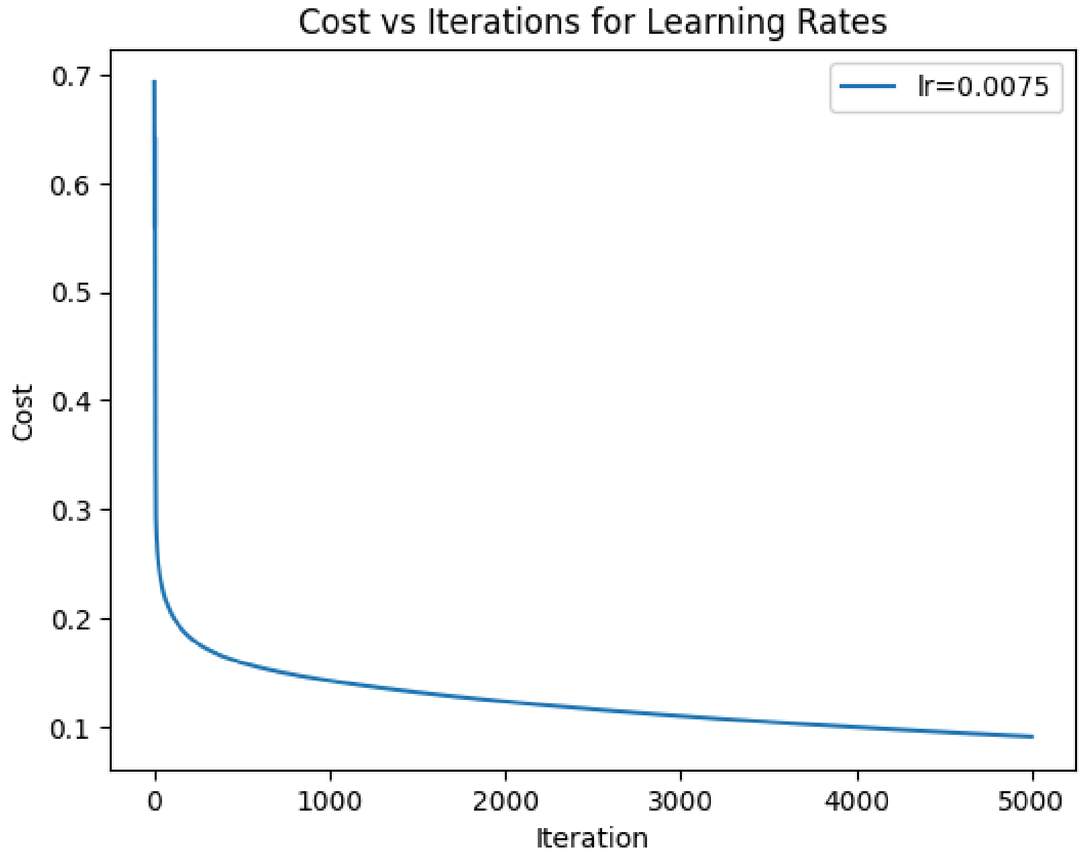
```
2 hidden layers → Accuracy: 95.00%
4 hidden layers → Accuracy: 77.29%
6 hidden layers → Accuracy: 62.50%
8 hidden layers → Accuracy: 50.00%
10 hidden layers → Accuracy: 50.00%
```



Logistic Regression - No hidden Layers

	Model	Final Training Accuracy	Final Validation Accuracy
0	model_lr_0.005	97.2222	93.1250
1	model_lr_0.05	99.7917	92.7083
2	model_lr_0.0684	99.8611	92.5000
3	model_lr_0.0075	97.5000	93.3333

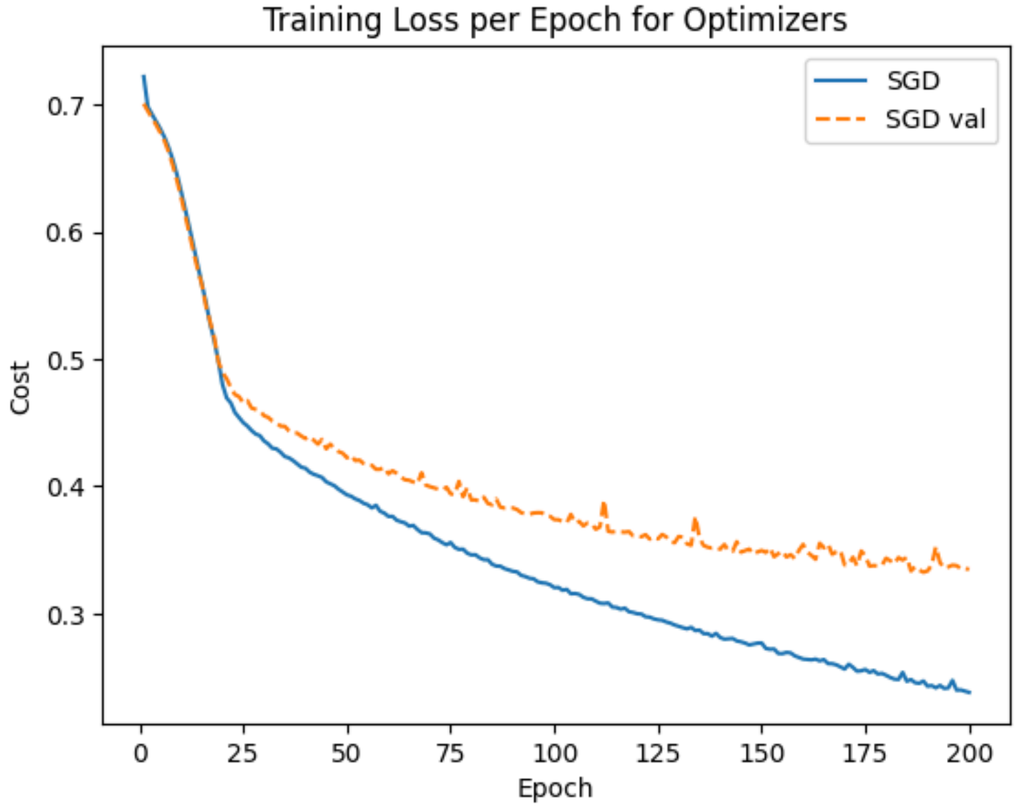
- **Most optimal Learning Rate: 0.0075**
- **0.05 & 0.0684 showing overfitting**



Optimization Algorithm

	Model	Final Training Accuracy	Final Validation Accuracy
0	model_opt_SGD	95.5556	93.1250
1	model_opt_RMSprop	49.1667	50.0000
2	model_opt_Adam	99.8611	92.9167

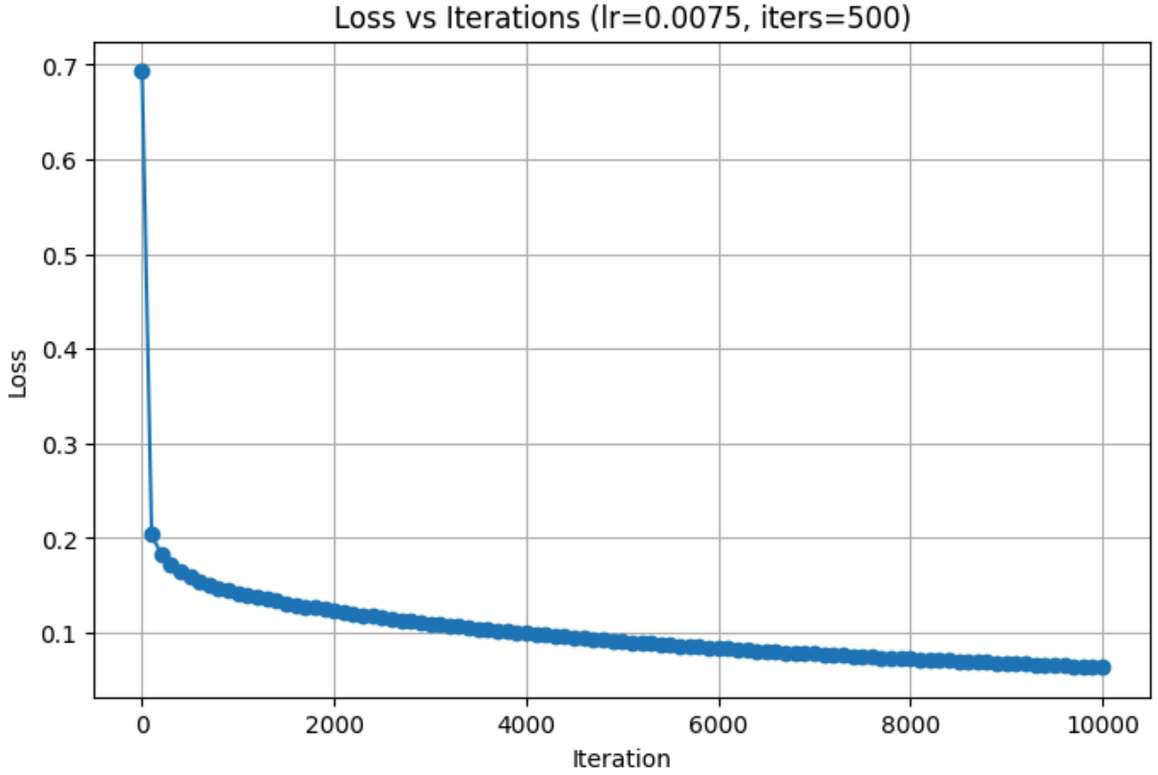
- **Most optimal Algorithm: SGD**
- **Smooth pattern with high accuracy**
- **RMSprop - no meaningful pattern**
- **Adam - Overfitting**



Epochs

- **Most optimal Epoch: 500**
- **Highest Validation Accuracy**
- **After 1000 epochs, as Epochs increases, more overfitting shown**

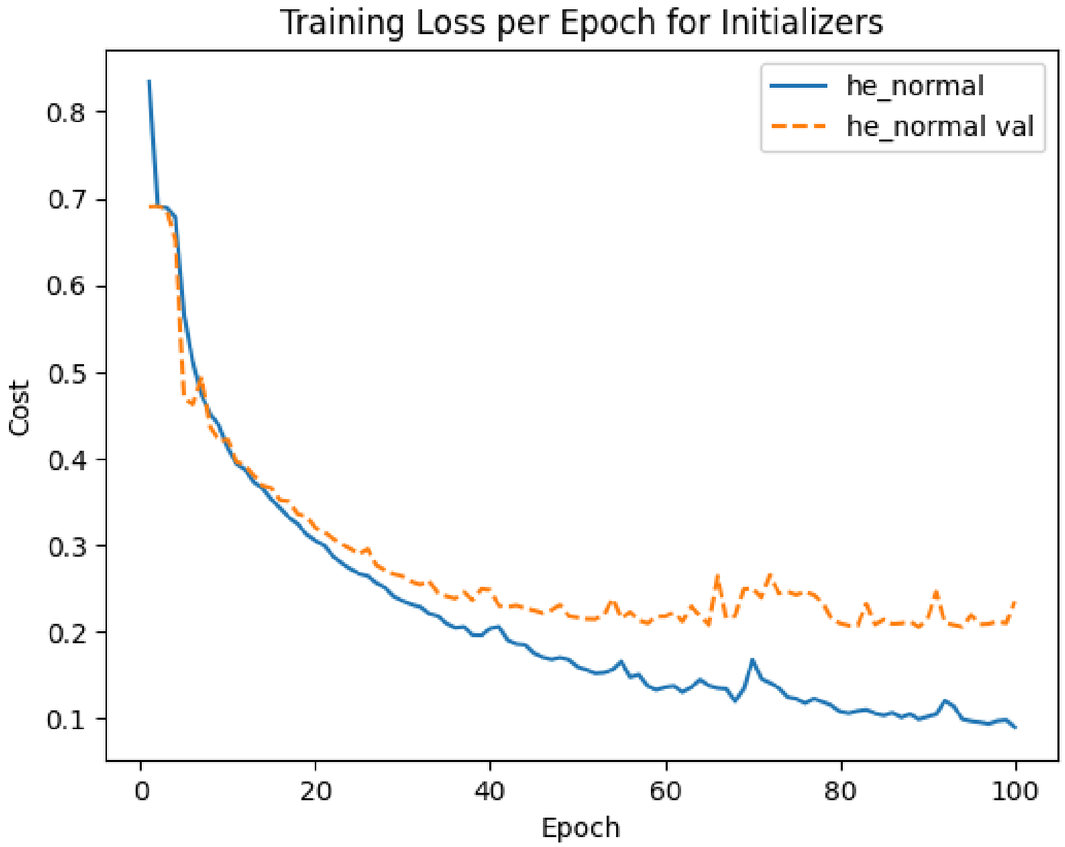
	Model	Final Training Accuracy	Final Validation Accuracy
0	model_epoch_100	92.9167	93.7500
1	model_epoch_500	95.1389	93.9583
2	model_epoch_700	95.5556	93.7500
3	model_epoch_1000	95.9028	93.7500
4	model_epoch_2500	97.0833	93.3333
5	model_epoch_5000	97.5000	93.3333
6	model_epoch_10000	98.5417	93.5417



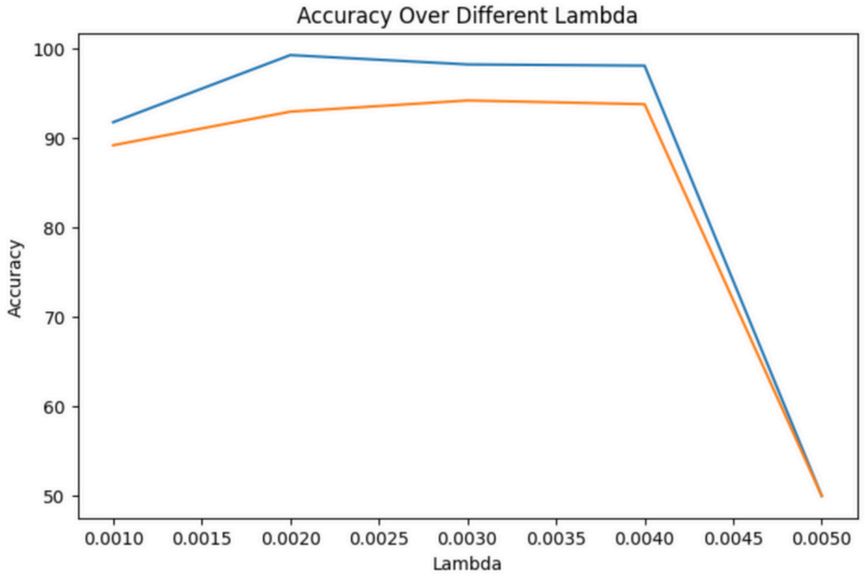
Weight Initializations

	Model	Final Training Accuracy	Final Validation Accuracy
0	glorot_uniform	0.9903	0.9312
1	he_uniform	0.9771	0.9333
2	he_normal	0.9819	0.9375
3	random_uniform	0.9903	0.9208

- **Most optimal Initialization: He normal**
- **He normal - minimal overfitting & smoothest descent & best generalization**



L2 Regularization



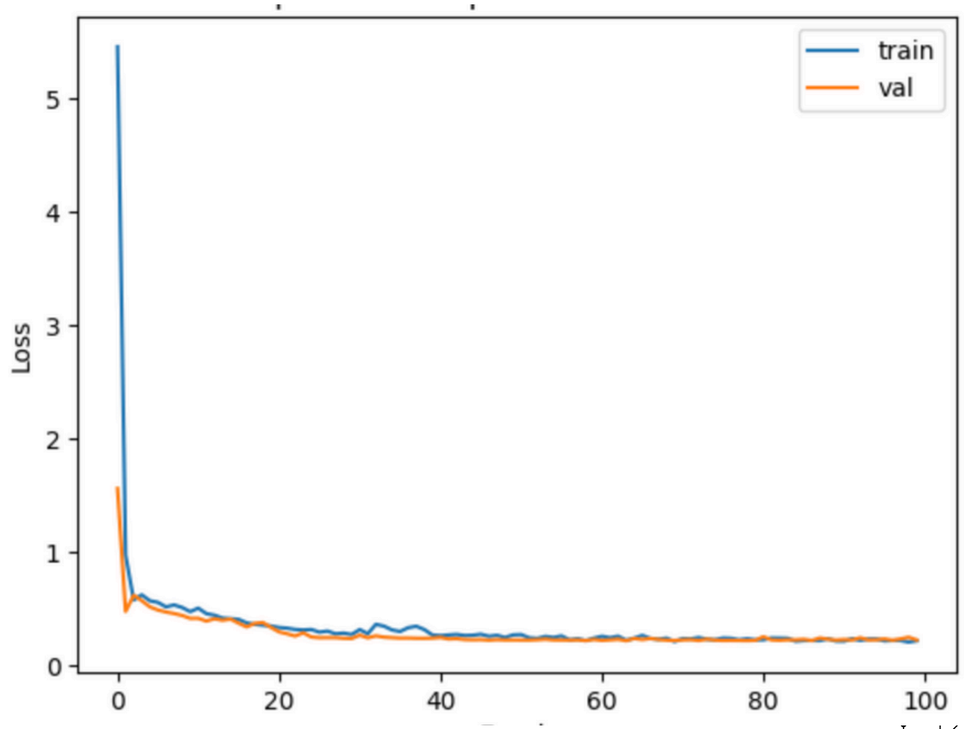
	Lambda	Training Accuracy (%)	Validation Accuracy (%)	Test Accuracy (%)
0	0.001	91.736111	89.166667	87.916667
1	0.002	99.236111	92.916667	91.875000
2	0.003	98.194444	94.166667	92.708333
3	0.004	98.055556	93.750000	91.666667
4	0.005	50.000000	50.000000	50.000000

L2 Rate: 0.1-->0.5

- **accuracy not improved much**
- **less overfitting(4% accuracy gap)**

Dropout Regularization

	dropout_rate	train_acc	train_recall	train_f1	val_acc	val_recall	val_f1
0	0.1	0.975000	0.979167	0.975104	0.925000	0.962500	0.927711
1	0.2	0.500000	1.000000	0.666667	0.500000	1.000000	0.666667
2	0.3	0.929167	0.984722	0.932895	0.927083	0.966667	0.929860
3	0.4	0.900000	0.809722	0.890076	0.925000	0.966667	0.928000
4	0.5	0.799306	0.998611	0.832658	0.941667	0.966667	0.943089
5	0.6	0.679167	0.986111	0.754516	0.902083	0.991667	0.910134
6	0.7	0.643750	0.993056	0.735975	0.912500	0.983333	0.918288
7	0.8	0.476389	0.620833	0.542476	0.500000	1.000000	0.666667



Dropout Rate: 0.1-->0.8

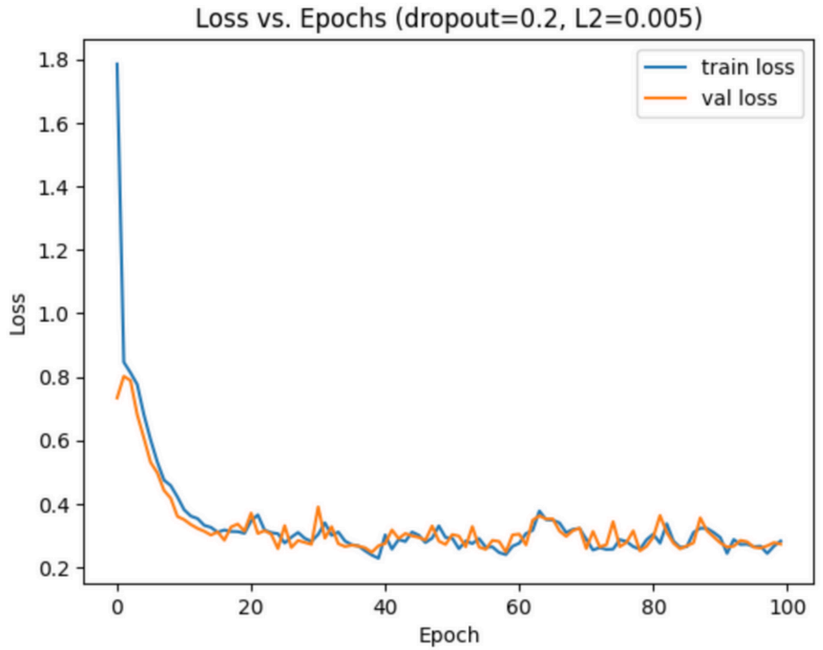
- **no severe overfitting problem as previous model(0.2% accuracy gap)**
- **validation recall is 96.67%--> good to minimize false negetives**

Dropout and L2 Regularization(mixture)

	dropout_rate	l2_rate	train_acc	train_recall	train_f1	val_acc	val_recall	val_f1
0	0.1	0.001	0.965278	0.952778	0.964838	0.931250	0.950000	0.932515
1	0.1	0.002	0.935417	0.962500	0.937120	0.925000	0.916667	0.924370
2	0.1	0.003	0.888889	0.877778	0.887640	0.939583	0.954167	0.940452
3	0.1	0.004	0.906944	0.945833	0.910428	0.891667	0.833333	0.884956
4	0.1	0.005	0.939583	0.941667	0.939709	0.885417	0.979167	0.895238
5	0.2	0.001	0.862500	0.970833	0.875940	0.933333	0.941667	0.933884
6	0.2	0.002	0.871528	0.968056	0.882837	0.935417	0.975000	0.937876
7	0.2	0.003	0.861111	0.972222	0.875000	0.941667	0.958333	0.942623
8	0.2	0.004	0.916667	0.936111	0.918256	0.931250	0.970833	0.933868
9	0.2	0.005	0.927778	0.958333	0.929919	0.931250	0.975000	0.934132
10	0.3	0.001	0.903472	0.959722	0.908613	0.891667	0.833333	0.884956
11	0.3	0.002	0.909722	0.956944	0.913793	0.937500	0.954167	0.938525
12	0.3	0.003	0.822222	0.984722	0.847073	0.935417	0.945833	0.936082
13	0.3	0.004	0.892361	0.943056	0.897554	0.935417	0.970833	0.937626
14	0.3	0.005	0.914583	0.923611	0.915348	0.912500	0.970833	0.917323

Dropout Rate: 0.1-->0.8
L2 Rate: 0.001-->0.005

- **high dropout--> underfitting**
- **recall is the highest 97.5%, little overfitting problem**



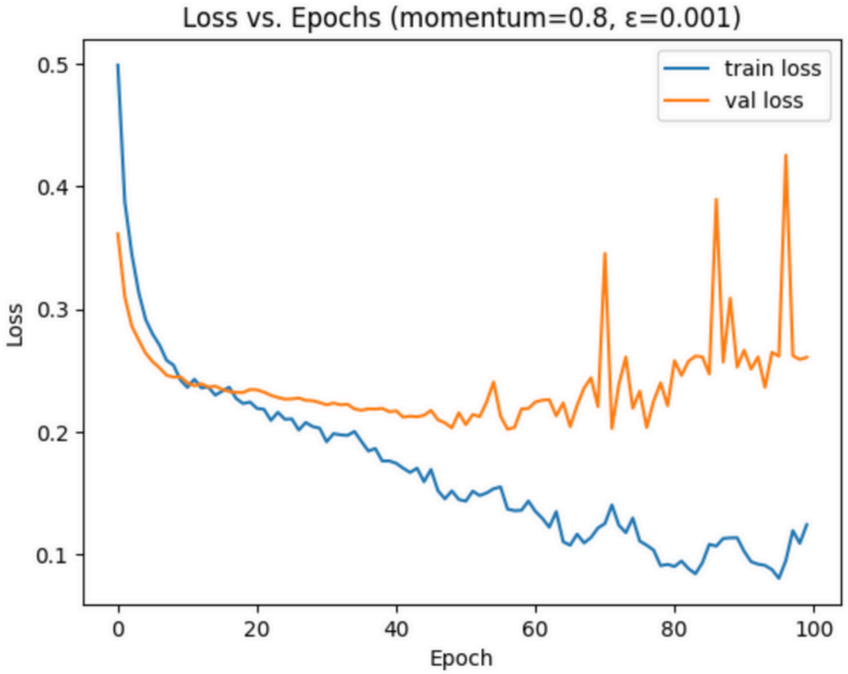
BatchNormalizations without Dropout

	momentum	epsilon	train_acc	train_recall	train_f1	val_acc	val_recall	val_f1
0	0.800	0.001000	1.000000	1.000000	1.000000	0.920833	0.933333	0.921811
1	0.900	0.000100	1.000000	1.000000	1.000000	0.927083	0.937500	0.927835
2	0.990	0.000100	0.995139	1.000000	0.995162	0.920833	0.937500	0.922131
3	0.900	0.000010	0.995833	0.998611	0.995845	0.927083	0.929167	0.927235
4	0.990	0.000010	0.995833	1.000000	0.995851	0.931250	0.933333	0.931393
5	0.999	0.000001	1.000000	1.000000	1.000000	0.714583	0.466667	0.620499

(Momentum, Epsilon) Combination:
(0.8, 1e-3), (0.9, 1e-4), (0.99, 1e-4), (0.9, 1e-5), (0.99, 1e-5), (0.999, 1e-6)

BatchNormalizations with Dropout(0.3)

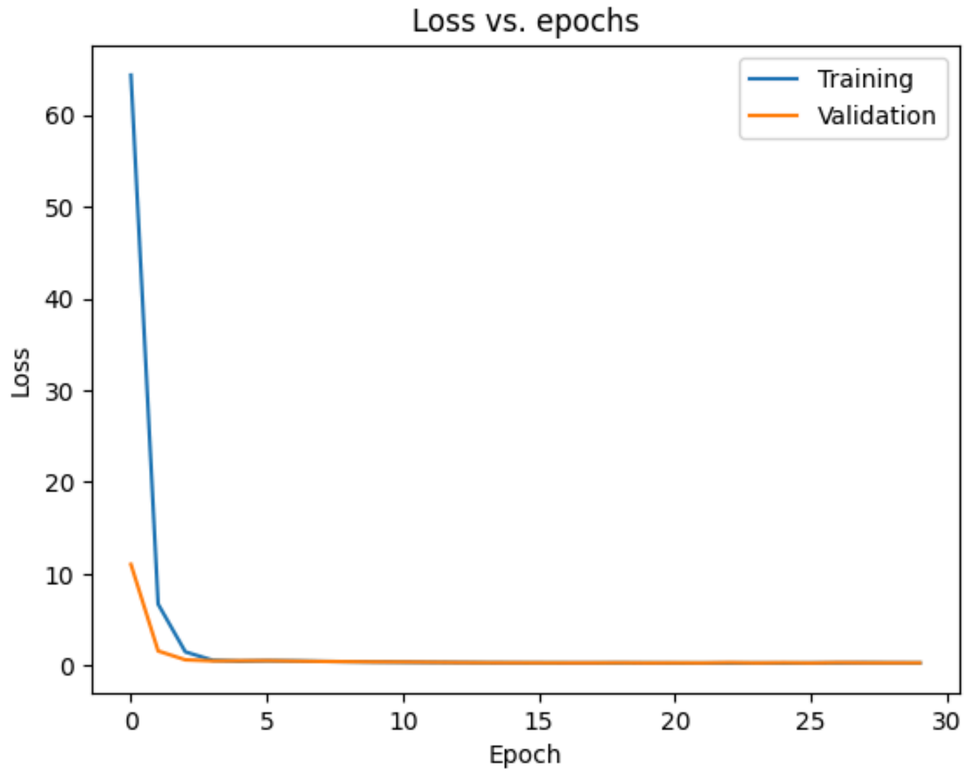
	momentum	epsilon	train_acc	train_recall	train_f1	val_acc	val_recall	val_f1
0	0.800	0.001000	0.956250	0.972222	0.956938	0.933333	0.970833	0.935743
1	0.900	0.000100	0.975000	0.973611	0.974965	0.929167	0.966667	0.931727
2	0.990	0.000100	0.979167	0.970833	0.978992	0.929167	0.970833	0.932000
3	0.900	0.000010	0.971528	0.986111	0.971937	0.918750	0.975000	0.923077
4	0.990	0.000010	0.961111	0.961111	0.961111	0.920833	0.933333	0.921811
5	0.999	0.000001	0.969444	0.966667	0.969359	0.770833	0.562500	0.710526



(Momentum, Epsilon) Combination:
(0.8, 1e-3), (0.9, 1e-4), (0.99, 1e-4), (0.9, 1e-5), (0.99, 1e-5), (0.999, 1e-6)
Dropout: 0.3 (best value or parameter in previous analysis)

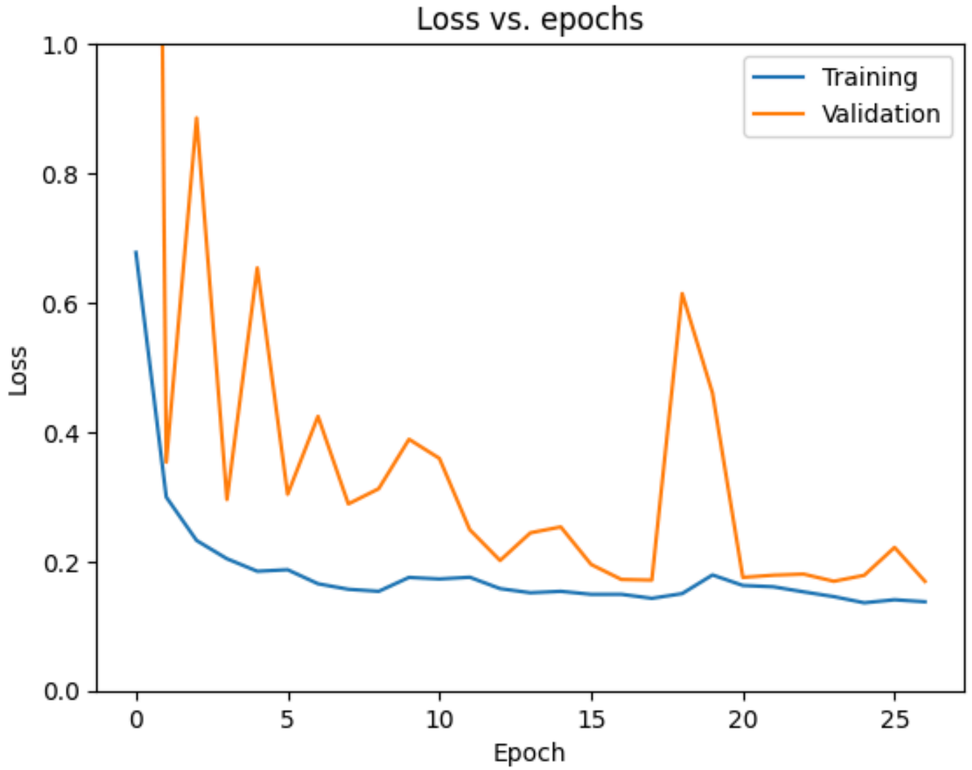
- **reletive high validation accuracy and high recall and less overfitting problem**

Early Stopping (Delta = 0.02, Patience = 10)



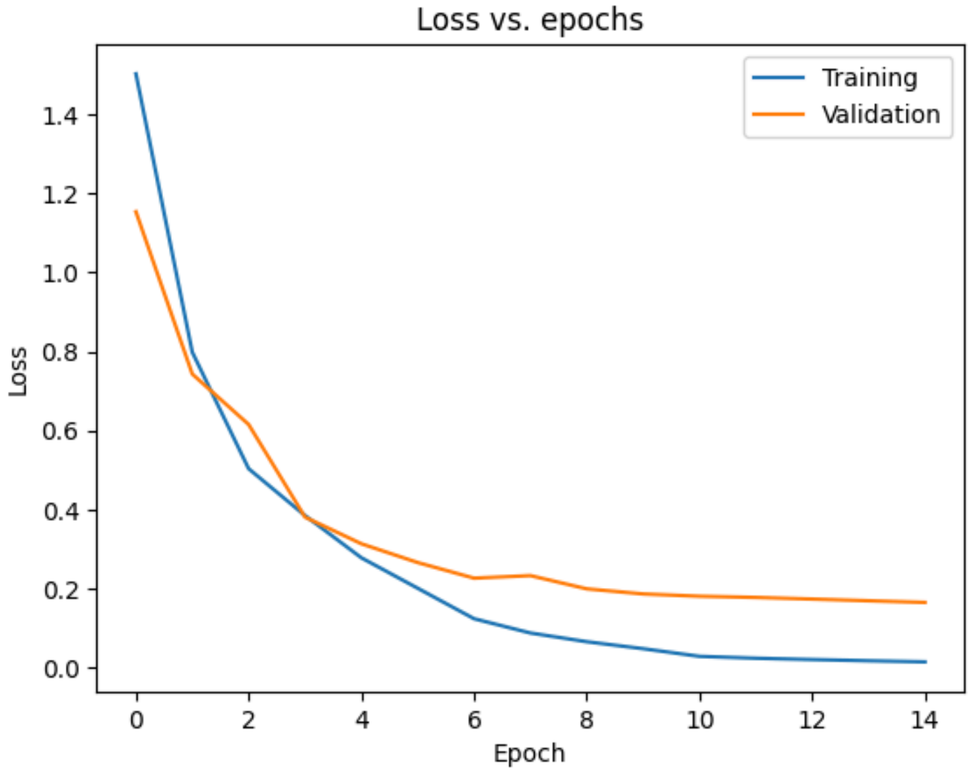
Monitoring Validation Loss
Final training accuracy: 0.8938, Final validation accuracy: 0.8958

Using CNN Model



Final training accuracy: 0.9572, Final validation accuracy: 0.9083

Pretrained Model: VGG



Final training accuracy: 0.9993, Final validation accuracy: 0.9438

Using RNN Model

Model	Training Accuracy	Validation Accuracy
Simple RNN	0.803472	0.820833
GRU	0.779167	0.806250
LSTM	0.659722	0.689583

Best Model

	momentum	epsilon	train_acc	train_recall	train_f1	val_acc	val_recall	val_f1
0	0.800	0.001000	0.956250	0.972222	0.956938	0.933333	0.970833	0.935743
1	0.900	0.000100	0.975000	0.973611	0.974965	0.929167	0.966667	0.931727
2	0.990	0.000100	0.979167	0.970833	0.978992	0.929167	0.970833	0.932000
3	0.900	0.000010	0.971528	0.986111	0.971937	0.918750	0.975000	0.923077
4	0.990	0.000010	0.961111	0.961111	0.961111	0.920833	0.933333	0.921811
5	0.999	0.000001	0.969444	0.966667	0.969359	0.770833	0.562500	0.710526

Validation Recall is the primary selection metric due to the critical need to detect all fire incidents.

Validation Result of Best Model

Training Accuracy	**95.63%**
Training Recall	**97.22%**
Training F1 Score	**95.69%**
Validation Accuracy	**93.33%**
Validation Recall	**97.08%**
Validation F1 Score	**93.57%**

High recall ensures fire cases are reliably detected in real-world scenario

Managerial Implications

01. Findings

Our optimized deep model (with Batch Normalization and Dropout) achieved better recall and F1 than logistic regression. It reduced false negatives significantly, which is critical in fire detection.

02. Why This Model

This model balances accuracy, robustness, and risk control.
It generalizes well after regularization, and it consistently outperforms simpler models on critical metrics like recall.

03. Business Value

Missing a fire can cost 500 dollars or more per case.
Reducing false negatives by 4 percent saves approximately 200,000 dollars per month, assuming 10,000 predictions.

04. Conclusion

The deep model is a better operational choice.
It is slightly more complex, but it minimizes risk, improves safety, and delivers greater long-term value.

Thank You

For Your Attention