COMPSCI-4AL3

Group 21 Milestone-3

Team

Name	MacID	Role
Suvansh Dutt	dutts1	Building the CNN and fine tuning the models.
Yuvraj Singh Sandhu	sandhy1	Data Loading and building the SVM.
Suhaas Parcha	parchas	Data Loading and building the KNN.

Context

a. What is the problem you have identified?

Residential fires are a significant environmental and societal issue, causing immense damage to ecosystems, properties, and human lives. It is crucial that we are able to detect fires as quickly as possible however with so many locations to cover and limited manpower we can only observe a limited region at any particular time. According to Federal Emergency Management Agency, the National Fire Incident Reporting System (NFIRS), and the National Safety Council: House fires cause an average of **2,620 civilian deaths** each year with home fires accounting for **92%** of civilian fire deaths (https://www.thezebra.com/resources/research/house-fire-statistics/). These also account for **$11.1 billion** in property damages every year.

b. Why is it a challenging problem?

Comparing CNN, KNN, and SVM on an image dataset is challenging due to their fundamentally different architectures and learning approaches. CNNs excel at automatic feature extraction through deep learning, while KNN and SVM rely on traditional machine learning techniques that require manual feature engineering. The key challenges include disparate computational requirements, complex hyperparameter tuning, varying performance metrics, potential overfitting, and the need to account for the specific characteristics of the image dataset. Each algorithm processes image data uniquely, making a fair and comprehensive comparison technically intricate and methodologically demanding.

c. How can it be solved through machine learning?

To address fire detection using machine learning, fine-tuning a CNN model will enable the model to learn the visual patterns associated with fires. Once trained, the model can analyze real-time CCTV footage, automatically detecting fires and alerting the fire department quickly, thereby enhancing response times and minimizing damage.

d. What aspects of the problem are you going to solve?

This project aims to develop an accurate and real-time fire detection system using machine learning, specifically focusing on analyzing CCTV footage to quickly identify fire incidents and alert the fire department. The key aspects to solve include image classification accuracy, real-time processing, and integration with existing CCTV infrastructure.

e. Why is it relevant to the world? How does it help

This fire detection system can save lives and reduce property damage by enabling rapid emergency response through automated AI-powered monitoring.

Dataset:

For our project, we used a publicly available dataset to train and evaluate our model. Below are the details of the dataset we utilized:

Fire Dataset - The dataset contains images related to fire and non-fire scenarios, which can be used to train the model for high-precision fire detection. The fire images depict residential fires, and the non-fire images are from diverse non-fire settings.

i. The dataset size is approximately 2 GB.
ii. Number of features:
    1. Fire images : 5853
    2. Non fire images : 9755
    3. Total - 15,608 features
iii. The dataset is hosted on kaggle and is publicly available. The link to the dataset is linked below
iv. Link to FIRE dataset

Proposed solution:

This project addresses a predictive problem of classifying images as fire or non-fire to aid fire departments. The features are image pixel values, with a binary fire/non-fire label as the target variable. We will compare three different machine learning techniques: CNN, SVM and KNN. CNNs have layers which are useful in predicting patterns. K-nn follows an easy implementation and is very robust to noise which is a very common byproduct of these datasets. Considering the complexity of an image base data, SVM's are a great model as well. Since, SVM and KNN are not able to handle high dimensional data quite well, they will be trained on PCA-reduced data. Preprocessing will involve image resizing, normalization, and PCA for dimension reduction. We will implement class balancing to ensure a balanced distribution between fire and non-fire images using techniques like oversampling or undersampling(less likely). We will use cross-validation, graphical analysis for evaluating our model to ensure robust model performance. The evaluation strategy will help in identifying if our model is overfitting or underfitting. The focus on recall ensures that the model minimizes false negatives, which is crucial in fire detection to avoid missing actual fire incidents. There are a few existing solutions available but those solutions use pre-trained weights and don't compare different classifications techniques. Libraries we plan to use for this project: Pytorch, numpy, sklearn, pandas, matplotlib, seaborn, pickle, PIL.

Preprocessing:

We applied several preprocessing techniques to ensure data readiness and improve model performance. These techniques include resizing, dimensionality reduction and standardization. We divided the data into 3 parts: train (70%), validation (15%) and test (15%).
For the CNN, the transformation applied on the inputs include a 128x128 resize to the inputs and normalization.
For both the SVM and KNN, the data was first resized to 64x64 and then standardized to have a mean of 0 and standard deviation of 1. Later, reduced the input features dimensionality to match the models capability, this was achieved by applying PCA with a variance of 0.95. The dimensions were reduced from 12000(64*64*3) to 400 approx.

Preprocessing Improvements:

CNN: Various transformations like gaussianblur, random flips, affine, etc were added since our model was overfitting previously. Also, we changed the resize from 128 x 128 to 256 x 256.
SVM:
We tried different approaches to reduce data using PCA.

1. We tried different variance values from 0.9 to 0.975. Here are the results using the best parameters on svm:
    a. Variance: 0.9, dimensions reduced to 139, test accuracy: 91.29%
    b. Variance: 0.925, dimensions reduced to 229, test accuracy: 91.29%
    c. Variance: 0.95, dimensions reduced to 413, test accuracy: 92.06%
    d. Variance: 0.975, dimensions reduced to 869, test accuracy: 91.50%
2. We also tried different scalers. Here are the test accuracies using 0.95 variance.
    a. MinMaxScaler, dimensions reduced to 380, test accuracy: 90.95%
    b. MaxAbsScaler, dimensions reduced to 380, test accuracy: 91.55%
    c. StandardScaler, dimensions reduced to 413, test accuracy: 92.06%
    d. RobustScaler, dimensions reduced to 420, test accuracy: 89.8%

KNN:
We tried different variance including 0.9 to 0.95, we concluded that 0.95 variance was the best as it was around ~400 features which was perfect for the model and avoided overfitting.

Model Specifications:

- Supervised method, the CNN processes 128x128 RGB images through 4 convolutional layers with batch normalization, ReLU activation, and 2x2 max-pooling after each layer. The first layer has 32 filters, followed by 64, 128, and 256 filters in subsequent layers. The output is flattened and passed through a fully connected layer with 1024 units (ReLU) and a final Sigmoid layer for binary classification. Training was done with a batch size of 64, learning rate of 0.01, over 100 epochs, using Binary Cross-Entropy Loss (BCELoss) and SGD optimizer. Batch normalization was used for regularization to ensure stable training.

- Supervised method, the **KNN model** classifies data using the k-nearest neighbours approach, where predictions are determined by the majority label among the nearest k points. It uses **torch.cdist** to compute pairwise distances between input data and training data, with support for GPU acceleration via PyTorch. The **fit** method stores the training data and labels, while the **predict** method processes input in batches (default size = 32) and performs majority voting for inference. The number of neighbors (k) is a configurable hyperparameter, with a default value of 5. Model evaluation is done using accuracy, calculated as the proportion of correctly classified samples.
- Supervised method, the Support Vector Machine (**SVM**) model uses a Radial Basis Function (RBF) kernel to effectively differentiate between "fire" and "not_fire" images within a PCA-reduced feature space. Hyperparameters include a regularization parameter C = 1 and a kernel coefficient gamma = 0.0001. The regularization parameter C balances the trade-off between achieving a wide margin and minimizing classification errors, while gamma determines the influence of individual training examples on the decision boundary. The core training mechanism resides in the **fit** function of the **SVMClassifier** class, which orchestrates the learning process by utilizing the CVXOPT library to solve the dual optimization problem inherent to SVMs. This involves constructing the RBF kernel matrix, setting up the quadratic programming formulation, and determining the optimal Lagrange multipliers. Through this optimization, the **fit** function identifies the support vectors that define the decision boundary and calculates the bias term necessary for accurate classification. This comprehensive approach ensures that the SVM model robustly captures the underlying patterns in the data, facilitating reliable and precise predictions on unseen images.

Model Specifications Improvements:

CNN:

- Overfitting was reduced by trying and adjusting parameters. Learning rate was changed from 0.01 to 0.02 (because of increased batch size), batch size from 64 to 200 (To utilize more GPU VRAM). The model was already at 96-97% test acc and these were done to reduce overfitting.

SVM:

- The SVM model was trained using different regularization parameter (C), kernel coefficient (gamma), and kernel type and evaluated on the same test set. The best performing set of parameters is **C** = 5, **Gamma** = 0.0001, **Kernel** = rbf with accuracy = 92.06, precision = 93, recall = 95, f1-score = 94.

KNN:

- The regular KNN was transformed into a weighted version, where the influence of each neighbor was determined by its distance to the query point. Closer neighbors were given higher weights, ensuring more significant contributions to the prediction. This enhanced accuracy by prioritizing relevant neighbors. It peaked at k = 3, TrA. = 99.7, TeA. = 87.14

Evaluation:

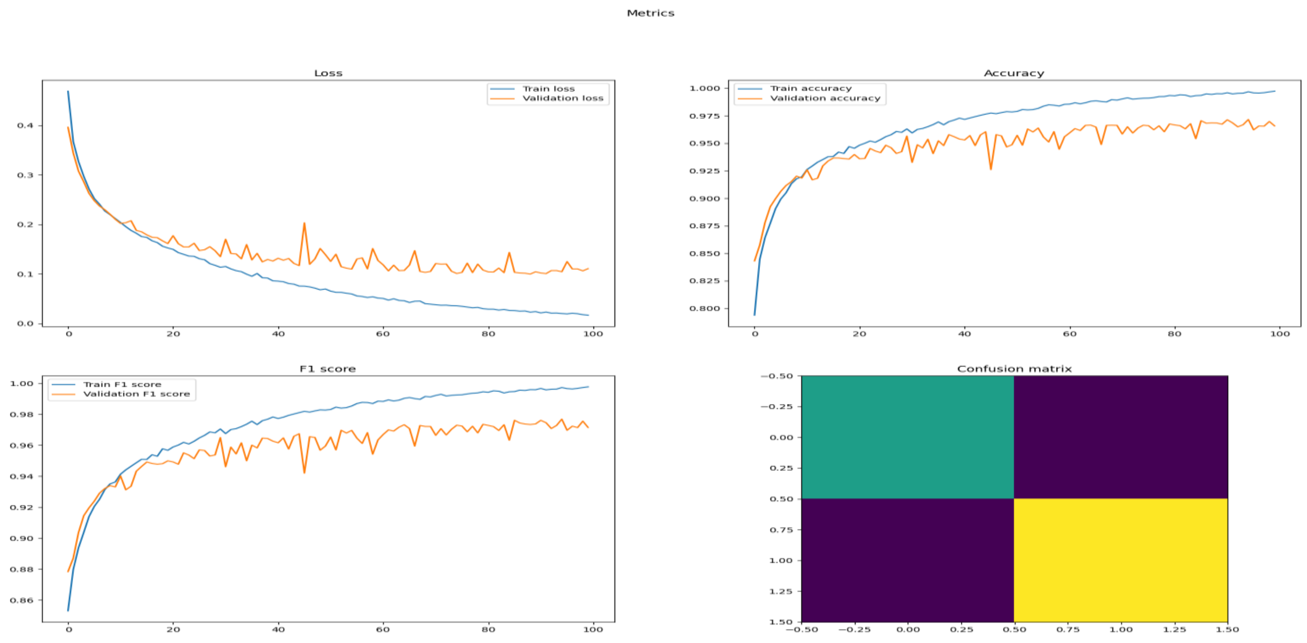
a. Evaluation During Training

- Train-Test Split: 70% training, 15% validation, 15% testing.
 - Metrics: - CNN: Training and validation loss per epoch;
. - KNN/SVM: Validation accuracy used to tune hyperparameters (k for KNN, C and gamma for SVM). -

b. Evaluation During Testing

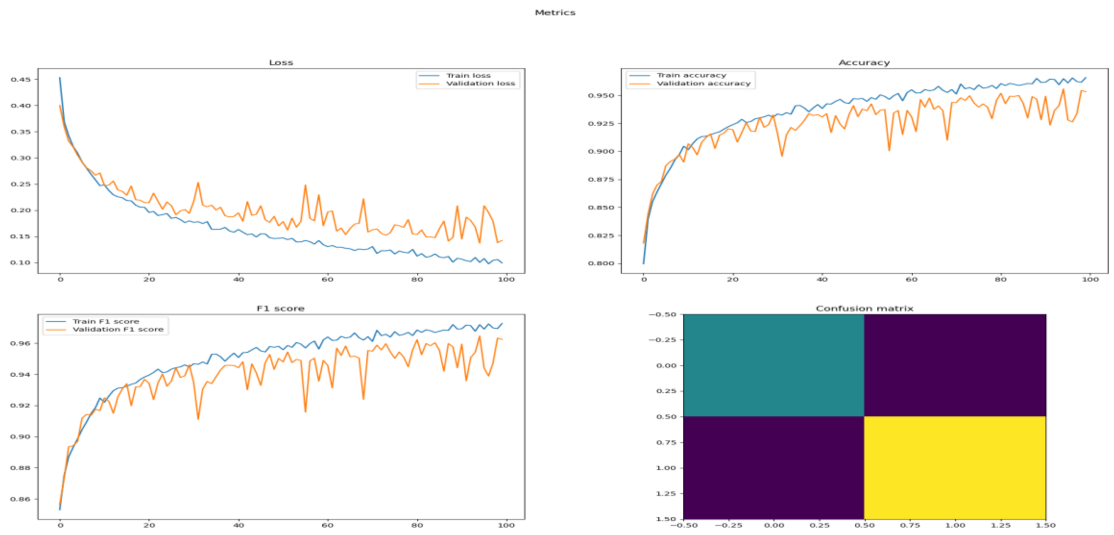
- Metrics:
- Accuracy: Proportion of correctly classified samples.
- Precision: True positives among all positive predictions.
- Recall: True positives among all actual positives.
- F1-Score: mean of precision and recall.
- Benchmarking: Compare CNN, KNN, and SVM based on these metrics.

Model	Train acc.	Test acc.	Loss.	precision	recall	f1
CNN	99%	97%	0.104	-	-	0.97
KNN(k=5)	90.11%	85.14%	-	0.89	0.91	0.91
SVM	94.5%	90.3%	-	0.91	0.94	0.92



Changes made after milestone 2:

Model	Train acc.	Test acc.	Loss.	precision	recall	f1
CNN	96%	94.6%	0.104	-	-	0.95
KNN(k=3)	99.7%	87.14%		0.88	0.92	0.91
SVM	99.8%	92.06%	-	0.93	0.95	0.93



Limitations:

Our preliminary results align with our expectations, as the CNN model outperforms KNN and the SVM in accuracy, precision, and recall due to its superior ability to extract complex features from high-dimensional image data. This validates our hypothesis that CNN would be the most effective model for fire detection in this project.

References:

Pytorch docs : https://pytorch.org/docs/stable/index.html
KNN - https://medium.com/@lakshmiteja.ip/understanding-weighted-k-nearest-neighbors-k-nn-algorithm-3485001611ce