Venue Calissification Using Decision Tree and CNN

Matin Mansouri
Master of Engineering Student
Concordia University
Montreal, Canada
matin.mansouri23@gmail.com

Zahed Ebrahimi
Master of Science Student
Concordia University
Montreal, Canada
zahedebrahimi89@gmail.com

Samane Vazrrian
Master of Science Student
Concordia University
Montreal, Canada
samane.vazirian@gmail.com

***Abstract*— This report presents a comparative study on venue classification using the StandardPlace365 dataset, which comprises five classes. The study investigates the performance of three models: a supervised Decision Tree, a semi-supervised Decision Tree, and a supervised Convolutional Neural Network (CNN) implemented in PyTorch. The dataset was preprocessed and split into training, validation, and test sets automatically. Each model was trained and evaluated using scikit-learn and PyTorch. The results demonstrate that CNN outperforms both Decision Tree models, achieving an accuracy of %71.20 compared to about 31% and 33% for the Decision Tree supervised and ssemi supervisedmodels on test data. By reading this report, one can expect a detailed explanation of the problem, the methodologies employed, and a comprehensive analysis of the experimental results, showcasing the CNN's superior classification capability for the given dataset.**

Keywords*— *supervised decision tree, semi-supervised decision tree, convolutional neural network, CNN

I. Introduction (*Heading 1*)

In machine learning and artificial intelligence, venue classification using image data is a significant challenge and opportunity [1].

This classification can be applied to numerous applications, including automated tagging in photo management software, enhancing search engine accuracy, and improving the user experience in location-based services [2].

The main objective of this project is to classify images into one of five distinct venue categories: airport terminal, movie theater, market, museum, and restaurant. To address this problem, we have employed decision tree models in both supervised and semi-supervised settings and a Convolutional Neural Network (CNN) implemented in PyTorch. Decision trees offer a simple and fast method for classification tasks, while CNNs are known for their superior performance in image recognition tasks due to their ability to capture spatial hierarchies in data.

Our approach involves training and evaluating these classifiers on a sampled subset of the StandardPlace365 dataset. As the original dataset was too large, with 365 classes, we selected a subset of the original dataset, including five classes. This subset comprises over 2,500 images, with each class containing approximately 500 images. The data has been split into training, validation, and test sets.

A key challenge in this problem is the model's tendency to overfit the training data, leading to high accuracy on the training set but significantly lower accuracy on the validation set. To tackle this, we explore various hyperparameter tuning techniques by adjusting parameters such as min_samples_split, min_samples_leaf, and max_depth for the decision tree model, and learning rate, batch size, and number of epochs for the CNN model. Additionally, we implemented semi-supervised learning techniques to leverage the unlabeled data. The comparative analysis of these models shows that CNN significantly outperforms the decision tree models, achieving an accuracy of about 69% compared to about 34% for the decision tree models.

II. Methodologies

This section outlines the comprehensive process of dataset preparation, model selection, and training for venue classification. We utilized a subset of the Places365Standard dataset [1], selecting five distinct venue classes. The dataset is balanced. We implemented three different models: a supervised Decision Tree, a semi-supervised Decision Tree, and a Convolutional Neural Network (CNN) using Sklearn and PyTorch. In this section, we aim to present the architecture and training process of each model, along with the hyperparameter tuning and optimization techniques used to achieve the best performance. The goal is to evaluate each model and compare the effectiveness of these models in accurately classifying venue images.

A. Dataset

The dataset used in this project is part of the publicly available Places365Standard dataset. It is a large-scale venue classification dataset, consisting of approximately 1.8 million training images and 36,500 validation images spread across 365 classes. The original images are colorful, 256x256 pixels in size, and can be downloaded from the official Places website [3]. These images are organized into folders based on their respective classes and are in JPEG format. For this project, a subset of the original dataset was selected, which includes five specific classes: 'airport_terminal', 'market', 'movie_theater', 'museum', and 'restaurant'. From each class, 500 images were randomly chosen to create a balanced dataset. This dataset was then divided into training, validation, and test sets with a split ratio of 64%, 16%, and 20%, respectively.

- Training Set: Contains images used for model training.
- Validation Set: Contains images used to evaluate model performance and tune hyperparameters.
- Test Set: Contains images used to evaluate the model using unseen data.

To prepare the data, we follow these steps. First, we load each image and resize it to 256x256 pixels to ensure all images are gathered at the same size. Then, we normalize the pixel values to the range [0, 1] by dividing them by 255.0. Finally, we flatten the images by converting them from 4D arrays (width, height, 3channels RGB) to 1D arrays to make them compatible with the decision tree classifier. Additionally, we encoded class labels, represented as folder names, int numerical values using the LabelEncoder from Sklearn. This process transformed categorical labels into integers.

In decision tree model, we use the flattened image pixels as features. Each pixel's value serves as an individual feature, resulting in a high-dimensional feature space that the decision tree can utilize for classification.

B. Decision Tree Model

Two types of Decision Tree models are implemented:

- Supervised Decision Tree: Trained solely on labeled data. We utilized the DecisionTree Classifier from the Sklearn library with the configuration's criterion='entropy' to assess the quality of splits. Then we change the hyperparameter to improve the model.
- Semi-supervised Decision Tree: Utilized both labeled and unlabeled data through a pseudo-labeling approach. Initially trained on a small portion of labeled data, the model iteratively labeled high-confidence predictions from the unlabeled data and retrained, progressively enhancing its performance.

C. CNN Model

For the CNN model, we designed a Convolutional Neural Network architecture using PyTorch. CNN was trained on the same dataset, utilizing various layers such as convolutional layers, pooling layers, and fully connected layers to extract features and perform classification. We performed hyperparameter tuning on parameters like learning rate, batch size, and the number of epochs to optimize the model's performance. To train the CNN, we used the Stochastic Gradient Descent (SGD) optimization algorithm. our final CNN model includes six convolutional layers for detailed feature extraction, a dropout rate of 0.2 to prevent overfitting, and a batch size of 64 for stable and efficient training. The fully connected (FC) section included four hidden layers, allowing the model to process the high-level 16384 features extracted by the convolutional layers.

The models were evaluated based on metrics, including precision, recall, and F1-score and confusion matrix.

D. Optimization Algorithm

In decision tree models, for hyperparameter tuning and determining the best hyperparameters, including max_depth, min_samples_split, and min_samples_leaf the GridSearchCV API from Sklearn is used. GridSearchCV uses a grid of potential values and applies cross-validation (CV) to assess the model's performance for each combination. The top-performing model is chosen based on validation accuracy. In our case, we applied GridSearchCV with a cross-validation (CV) set to 5.

In optimizing our CNN model, we experimented with various parameters to enhance its performance. It included convolutional (CONV) layers, learning rate, batch size, optimization method hidden layers in fully connected (FC) classifiers, and dropout parameters.

We expanded the model from 4 to 6 convolutional layers by increasing filter sizes from 32 to 1024, each followed by ReLU activation, and batch normalization. Also, we remove and add the pooling. We tested optimizers, including Adam and Stochastic Gradient Descent (SGD) with momentum, finding that Adam provided better stability and faster convergence. For batch sizes, 64 proved optimal, balancing performance and training stability. In further refining feature extraction, our fully connected (FC) classifier included six layers, efficiently reducing the feature space to the final classification output. Finally, a dropout rate of 0.2 was selected to effectively prevent overfitting, achieving robust performance. These combined optimizations led to our final CNN model, which excelled in both validation and test datasets.

III. Result

This section details the experimental design, optimization strategies, and the performance outcomes of our models. We structured the analysis into three main parts: Experiment Setup, Main Results, and Ablative Study.

A. Experiment Setup

For our experiments, we utilized a subset of the Places365Standard dataset, focusing on five venue categories: airport terminal, market, movie theater, museum, and restaurant. Each category comprised around 500 images, which were divided into 64% for training, 16% for validation, and 20% for testing. We resized and normalized the images to 256x256 pixels. The CNN architecture included six convolutional layers with progressively increasing filter sizes and a fully connected section of six layers. Throughout the optimization process, we experimented with adding or removing pooling layers, adjusting the number of convolutional layers, and fine-tuning hyperparameters such as learning rates, batch sizes, and dropout rates. Data augmentation techniques were applied to improve model generalization. Performance was evaluated using accuracy, precision, recall, and F1-score, confusion matrix metrics.

B. Main Results

The main results are presented in terms of accuracy, precision, recall, and F1-score across the validation and test datasets for each model.

1) Supervised Decision Tree: Initial attempts with default hyperparameters resulted in high training accuracy but poor validation performance due to overfitting (31% validation accuracy). Optimizing max_depth, min_samples_split, and min_samples_leaf improved validation accuracy marginally to 36%. Test performance showed an accuracy of 31%, confirming the model's limited generalization capability.

2) Semi-supervised Decision Tree: The pseudo-labeling approach slightly improved performance, achieving 32% validation accuracy and 33% test accuracy, but still lagged behind CNN.

3) Convolutional Neural Network (CNN): Initial Model Achieved a validation accuracy of 51.5%, significantly outperforming the Decision Tree models. Final Model with six convolutional layers, a dropout rate of 0.2, and a batch size of 64, the CNN reached a validation accuracy of 74.50% and a test accuracy of 70.00%. These results highlight CNN's superior ability to learn and generalize from complex image data. The results can be found in Table I.

C. Ablative Study

*a) Supervised Decision Tree:*In the first attempts, we ran the DecisionTree Classifier from scikit-learn with default hyperparameters (criterion='entropy'. The model achieved perfect accuracy on the training data but showed significant overfitting with a validation accuracy of only 31% (Table II

and Table III). This disparity highlighted the model's lack of generalization capability.

Hyperparameter Tuning: To mitigate overfitting, we adjusted several hyperparameters, including max_depth, min_samples_split, and min_samples_leaf. Using GridSearchCV, we systematically examined the effects of these parameters.

TABLE I. COMPARE THE FINAL MODEL ACCURACY ON TEST AND VALIDATION DATA ON MODELS, INCLUDING SUPERVISED DECISION TREE, SEMI-SUPERVISED DECISION TREE AND CNN IN TWO PHASE INITIAL AND AFTER OPTIMIZATION.

Model	Validation Accuracy	Test Accuracy	Recall (Validation)	Recall (Validation)	F1 Score (Validation)
Supervised Decision Tree (initial)	31%	29%	0.31	0.31	0.31
Supervised Decision Tree(final)	36%	31%	0.36	0.35	0.35
Semi-supervised Decision Tree	32%	33%	33.00	0.32	0.32
CNN (Initial)	51.50%	52.60%	0.53	0.51	0.49
CNN (Final)	74.50%	70%	0.75	0.74	0.74

TABLE II. METRICS PERFORMANCES FOR TRAIN AND VALIDATION DATA WITH DEFAULT HYPERPARAMETER AND CRITERION='ENTROPY'

Metrics value for Train Data				
Metric	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Value	100%	1.0	1.0	1.0
Metrics value for Validation Data				
Metric	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Value	31%	0.31	0.31	0.31

TABLE III. CONFUSION MATRIX FOR TRAIN AND VALIDATION DATA WITH DEFAULT HYPERPARAMETER AND CRITERION='ENTROPY'

Predicted / True	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*
Class 1	26	15	12	20	7
Class 2	15	14	5	25	21
Class 3	2	12	43	7	16
Class 4	14	17	8	22	19
Class 5	11	20	15	15	19

The best configuration (max_depth=10, min_samples_split=20, min_samples_leaf=13) improved the balance between training and validation performance, reducing overfitting (Table IV and Table V). Despite these optimizations, the validation accuracy increased only marginally to 36%, indicating room for further improvement.

Test Performance: When evaluated on the test set, the model with optimized hyperparameters demonstrated an accuracy of 31%, confirming the challenges in achieving robust performance across unseen data (Table VI and Table VII).

TABLE IV. METRIC VALUES FOR TRAINING AND VALIDATION DATA WERE OBTAINED USING CRITERION='ENTROPY' WITH VARIOUS DP, SP, AND LE

Training - Validation				
Dp, Sp, Le	*Accuracy*	*Precision*	*Recall*	*F1 Score*
10, 15, 13	0.73-0.36	0.73-0.35	0.73-0.36	0.73-0.35
10, 20, 13	0.73-0.36	0.73-0.36	0.73-0.36	0.73-0.36
10, 30, 13	0.71-0.36	0.71-0.35	0.71-0.36	0.71-0.35
10, 20, 8	0.79-0.35	0.79-0.35	0.79-0.35	0.79-0.35
10, 20, 11	0.75-0.35	0.75-0.35	0.75-0.35	0.75-0.35
8, 20, 13	0.66-0.35	0.66-0.35	0.66-0.35	0.66-0.35
12, 20, 13	0.74-0.38	0.74-0.38	0.74-0.38	0.74-0.38

TABLE V. CONFUSION MATRIX FOR BEST ACCURACY FOR HYPER PARAMETER DP = 10, SP = 20, LE = 13

Predicted class / True class	Train					Validation				
	1	2	3	4	5	1	2	3	4	5
1	241	28	14	19	18	28	17	8	15	12
2	25	229	7	36	23	16	22	9	23	10
3	22	20	271	4	3	7	7	50	6	10
4	17	25	17	242	19	21	18	8	26	7
5	38	25	43	30	184	14	23	11	13	19

b) Semi-supervised Decision Tree: Semi-supervised Approach: We implemented a pseudo-labeling strategy to leverage unlabeled data. This method involved using the trained model to predict labels for the unlabeled set and iteratively adding high-confidence predictions (confidence > 0.9) back into the training set. Despite this effort, the semi-supervised model's performance did not significantly exceed that of the purely supervised model. The validation accuracy was slightly improved to 32%, and the test accuracy to 33% (Table VIII and Table IX).

TABLE VI. TEST DATA METRIC PERFORMANCE OF BEST TABLE III

Metrics value for Test Data				
Metric	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Value	31%	0.3	0.31	0.3

TABLE VII. TEST DATA CONFUSION MATRIX OF BEST TABLE IV

Predicted / True	*Class 1*	*Class 2*	*Class 3*	*Class 4*	*Class 5*
Class 1	38	13	10	22	17
Class 2	28	16	20	20	16
Class 3	13	14	55	7	11
Class 4	24	20	13	28	15
Class 5	20	17	25	22	16

TABLE VIII. VALIDATION AND TEST METRIC VALUES FOR SEMISUPERVISED DECISIONTREECLASSIFIER WITH DP = 10, SP= 20, LE = 13

Metrics value for Validation Data				
Metric	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Value	32%	0.31	0.32	0.31
Metrics value for Test Data				
Metric	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Value	33%	0.32	0.33	0.32

TABLE IX. CONFUSION MATRIX FOR VALIDATION AND TEST DATA FOR SEMI-SUPERVISED DECISION TREE CLASSIFIER WITH DP = 10, SP= 20, LE = 13

Predicted class / True class	*Validation*					*Test*				
	1	2	3	4	5	1	2	3	4	5
1	19	12	4	15	10	37	21	14	20	12
2	12	15	8	9	11	12	22	19	18	16
3	2	1	21	4	7	9	8	52	5	18
4	7	11	6	18	8	20	14	19	28	26
5	11	10	14	8	7	20	20	27	16	27

c) Convolutional Neural Network (CNN): In our initial attempt to create a CNN model, we designed a structure comprising four convolutional layers. Each layer was followed by batch normalization and max pooling.

Conv1 used 32 3x3 filters on the input image with padding to maintain spatial dimensions. Conv2, connected to the first layer, applied 64 3x3 filters to further refine the learned features. For Conv3, we utilized 128 3x3 filters, and for Conv4, we employed 256 3x3 filters to extract deep and detailed features. After each convolutional layer, a batch normalization layer was included to stabilize and speed up the training process, along with a max pooling layer to reduce spatial dimensions and focus on the most significant features.

After the convolutional and pooling layers, the model utilized fully connected layers to transform the learned feature space into output classes. In the FC1 layer, the flattened vector was converted into 512 neurons to capture a dense representation of the high-dimensional input space. Then, in FC2, the number of neurons was reduced to 128 to create a more distinct representation of the learned features. Finally, FC3 mapped the 128 neurons to the 5 output classes corresponding to the venue categories. These fully connected layers involved ReLU activations. Also, we employed data augmentation techniques such as random horizontal flips, rotations, and color adjustments on the training data to enhance the model's ability to generalize.

In the initial training, the model achieved a validation accuracy of 51.5%, showing a significant improvement over the Decision Tree models. However, it also pointed out the need for further refinement to enhance. The results were shown in Table X.

1) Adding Hyperparameter Tuning and Optimization Strategies: Our base CNN model was trained without a standard optimizer, relying on manual learning rate adjustments. To improve the model's performance, we introduced the Adam optimizer, which adjusts learning rates dynamically based on gradient estimates. Adam significantly enhanced training stability and convergence speed, boosting the validation accuracy to 61.00%. This improvement is attributed to Adam's ability to provide smoother and more consistent gradient updates, reducing the risk of getting stuck in local minima.

TABLE X. PERFORMANCE ON INITIAL MODEL WITHOUT ANY OPTIMIZATION

Metrics value for Validation Data				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**51.50%**	**0.53**	**0.51**	**0.49**

The Stochastic Gradient Descent (SGD) optimizer with a momentum of 0.9 is tested, which helps accelerate gradient vectors in the correct direction, leading to faster convergence. Although SGD, with momentum, improved training stability and speed, it did not surpass the performance achieved with Adam. Adam's superior results are likely due to its combined advantages of RMSprop and AdaGrad, making it more effective for training deep networks. The results can be found in Table XI.

2) Adjusting Learning Rate: After introducing the Adam optimizer, we conducted experiments with various learning rates to determine the most effective value. We tested several learning rates, including .001,01,0001,00001. We found that the best result is with a learning rate of 0.0001, with an accuracy of 64.25% for validation data. The results, comparing different learning rates, can be found in Table XII and Fig. I.

TABLE XI. MODELS' PERFORMANCE WITH ADAM AND GSD OPTIMIZER

Metrics value for Validation Data (SGD)				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**61.00%**	**0.61**	**0.61**	**0.60**
Metrics value for Validation Data (ADAM)				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**56.00%**	**0.56**	**0.58**	**0.55**

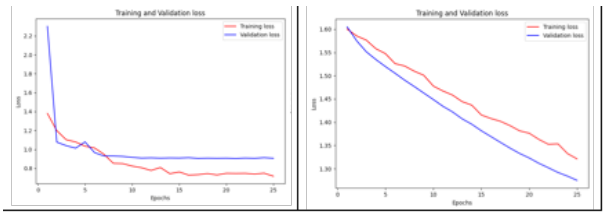

FIG. I. COMPARE LOSING FUNCTION BEFORE AND AFTER THE APPLING ADAM OPTIMIZER ON MODEL WITH 4 CONVOLUTIONAL LAYERS

TABLE XII. ADJUSTING LEARNING RATE WITH 0.0001, 0.0001, 0.001, 0.01, 0.1, AND BATCH SIZE = 64

Metrics value for Validation Data 0.00001				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**59.00%**	**0.59**	**0.59**	**0.58**
Metrics value for Validation Data 0.0001				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**64.25%**	**0.65**	**0.64**	**0.64**
Metrics value for Validation Data 0.001				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**57.00%**	**0.56**	**0.57**	**0.56**
Metrics value for Validation Data 0.01				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**56.00%**	**0.58**	**0.56**	**0.55**

3) Learning Rate and Batch Size: In this stage, we evaluated batch sizes of 32, 64, and 128 to assess their impact on the validation data. A batch size of 64 proved to strike a good balance, facilitating consistent gradient updates and contributing to the model's ability to generalize effectively. The results can be found in Table XIII.

4) Adding Additional Convolutional Layers: To further optimize and extract more complex features from the images, we experimented with increasing the number of convolutional layers in the network. By adding two, three, and four additional convolutional layers, we aimed to capture complex patterns and hierarchies in the data. The best results were achieved with six convolutional layers, by adding the first layer containing 32 filters of size 3x3 and the sixth layer containing 1024 filters of size 3x3. Deeper networks increased the network's capacity to learn complex representations, providing detailed and deep feature extraction capabilities. The accuracy for validation data was 64.25%. Additionally, Table XIV presents the accuracy-related models with depths of 4, 5, 6, and 7.

5) Adjusting Dropout Rate: Dropout is an essential technique to prevent overfitting in neural networks by randomly deactivating neurons during training. This process encourages the network to learn more robust and generalized features.

To determine the optimal dropout rate for our model, we experimented with two specific rates: 0.2 and 0.5, applied after the first fully connected layer. The dropout rate of 0.2 yielded the best performance, achieving an accuracy of 74.50% on the validation set, and 86.19% on the training set. These results indicate that a lower dropout rate allows the network to retain enough capacity for learning while still effectively mitigating overfitting. With a dropout rate of 0.5, the model achieved a validation accuracy of 66.75% and a training accuracy of 77%. While this higher dropout rate effectively reduced the training sets' overfitting, it also slightly diminished the network's ability to learn complex patterns, leading to lower performance on both the training and validation sets.

These findings suggest that a dropout rate of 0.2 strikes a better balance between maintaining performance and preventing overfitting, allowing the network to generalize well to unseen data. The detailed performance metrics for these experiments are summarized in Table XV.

TABLE XIII. MODEL ACCURACY ON VALIDATION DATA WITH BATCH SIZE WITH LEARNING RATE 0.001, 0.0001 AND BATCH SIZE = 32, 64, 128

Batch Size	Learning Rate	
	0.001	0.0001
32	59.5%	66.5%
64	57%	64.25%
128	57.5	61%

TABLE XIV. ADDING ADDITIONAL CONVOLUTIONAL LAYERS (4, 6)

Metrics value for number of Convolutional Layer = 4				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**59.00%**	**0.59**	**0.59**	**0.58**
Metrics value for number of Convolutional Layer = 5				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**62.50%**	**0.63**	**0.62**	**0.62**
Metrics value for number of Convolutional Layer = 6				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**62.5%**	**0.63**	**0.62**	**0.67**
Metrics value for number of Convolutional Layer = 7				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**71.00%**	**0.71**	**0.71**	**0.71**

TABLE XV. ADJUSTING DROPOUT RATE THROUGH 0.2 FOR NUMBER OF CONV = 6, BATCH SIZE = 64 AND LR = 0.0001

Metrics value for validation data Dropout Rate=.2				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**74.50%**	**0.75**	**0.74**	**0.74**
Metrics value for train data Dropout Rate=.2				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**86.19%**	**0.86**	**0.86**	**0.86**
Metrics value for validation data Dropout Rate=.5				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**66.75%**	**0.6880**	**0.6675**	**0.6682**
Metrics value for train data Dropout Rate=.5				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**77.00%**	**0.7697**	**0.7700**	**0.769**

6) Adding Hidden Layers in Fully Connected (FC) Layers: To evaluate the impact of increasing the depth of the fully connected (FC) layers in our CNN model, we tested two different architectures: one with four hidden layers and another with five. The first model's architecture included layers that progressively reduced the neuron count from 1024 to 512, 256, and finally to 128 neurons before the output layer mapped these 128 neurons to the 5 output classes. In the second model, we added a hidden layer that reduced the neurons from 128 to 64 before the final classification. The results showed that the first model, with four hidden layers, achieved a validation accuracy of 71.25% and a training accuracy of 84.00%. The second model, with five hidden layers, improved the validation accuracy to 72.75%, although it slightly decreased the training accuracy to 81.88%, indicating a good balance between learning and generalization. These results suggest that adding more depth to the FC layers allows the model to capture and process more complex features, enhancing its overall performance. Detailed performance metrics are summarized in Table XVI.

TABLE XVI. ADJUSTING FC HIDDEN LAYERS FOR TWO AND THREE HIDDEN LAYERS

Metrics value for validation data with 2 hidden layers				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**71.25%,**	**0.72**	**0.71**	**0.71**
Metrics value for train data with 2 hidden layers				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**84.00%**	**0.83**	**0.84**	**0.83**
Metrics value for validation data with 3 hidden layers				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**72.75%,**	**0.73**	**0.72**	**0.73**
Metrics value for train data with 3 hidden layers				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**81.88%**	**0.81**	**0.81**	**0.81**

7) Final model: After extensive experimentation with various architectures and hyperparameters, our final CNN model achieved the best validation results. This optimized model includes six convolutional layers for detailed feature extraction, a dropout rate of 0.2 to prevent overfitting, and a batch size of 64 for stable and efficient training. The fully connected (FC) section included four hidden layers, allowing the model to process the high-level 16384 features extracted by the convolutional layers. This configuration led to a balanced and robust model, which performed well on both the validation and test datasets. On the test data, the model achieved an accuracy of 71.20%, reflecting its generalization capabilities and effectiveness in handling the venue classification tasks. The detailed performance metrics for this model are provided in Table XVII and Table XVIII and Fig. II.

TABLE XVII. METRICS VALUE FOR TEST DATA ON FINAL MODEL WITH BATCH SIZE = 64 NUMBER OF CONVOLUTIONAL LAYER = 6 OPTIMIZER = ADAM WITH 5 LAYERS FULLY CONNECTED CLASSIFIER WITH DROPOUT 0.2

Metrics value for test data				
Metric	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Value	**71.20%**	**0.72**	**0.71**	**0.71**

TABLE XVIII. CONFUSION MATRIX FOR TEST DATA ON THE FINAL MODEL

Predicted True	***Class 1***	***Class 2***	***Class 3***	***Class 4***	***Class 5***
Class 1	**75**	**7**	**3**	**8**	**7**
Class 2	**6**	**77**	**0**	**2**	**15**
Class 3	**3**	**2**	**82**	**4**	**9**
Class 4	**11**	**4**	**1**	**58**	**26**
Class 5	**13**	**6**	**8**	**9**	**24**

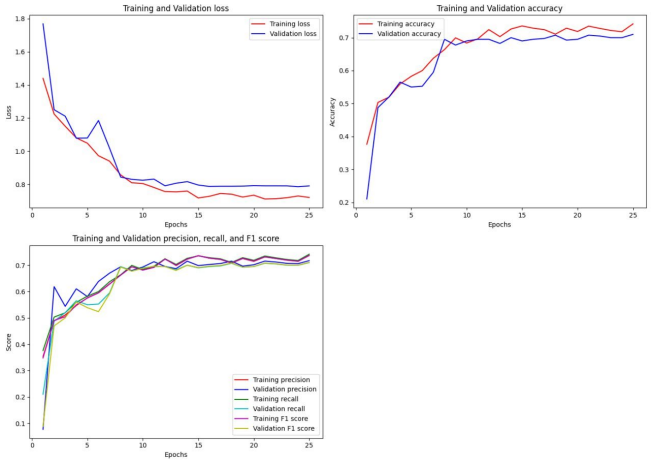

FIG. II. TRAINING AND VALIDATION LOSS, ACCURACY, AND PRECISION, RECALL, AND F1 SCORE ON FINAL MODEL

REFERENCES

[1] Shi, Hongju. "Application of artificial intelligence technology in the information management of sports venues." Revista Ibérica De Sistemas e Tecnologias De Informação 16B (2015): 150.

[2] Sen, Pratap Chandra, Mahimarnab Hajra, and Mitadru Ghosh. "Supervised classification algorithms in machine learning: A survey and review." Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018. Springer Singapore, 2020.

[3] Places365 website: http://places2.csail.mit.edu/download.html