Exploiting Local and Global Discriminative Information for Indoor Scene Recognition

Harmya Bhatt
Purdue University
West Lafayette, Indiana
hvbhatt@purdue.edu

Rohan Purandare
Purdue University
West Lafayette, Indiana
rpuranda@purdue.edu

Shruti Goyal
Purdue University
West Lafayette, Indiana
goyal67@purdue.edu

Arnav Arora
Purdue University
West Lafayette, Indiana
arora165@purdue.edu

ABSTRACT

Classifying indoor scenery remains a significant challenge in computer vision. While traditional Convolutional Neural Networks (CNNs) have significantly transformed conventional methodologies in this regard, our research aims to investigate the impact of the latest, state-of-the-art CNN models, with a specific focus on refining a distinctive dimension: object definition within the scenery. By incorporating object definition as a key feature in the training process, we have conducted a comprehensive benchmarking study, comparing the effect of traditional methods, CNNs, and using transfer learning from pre-trained CNNs against the progress made by more outdated techniques in this domain.

1 INTRODUCTION

The primary goal of this project is to conduct an exhaustive benchmarking study to examine the impact of various classification models and sampling strategies on key evaluation metrics such as accuracy, F1 score, and validation loss. Several recent approaches have attempted to improve indoor scene classification through the integration of relative spatial relationships and object identification. Our initial approach was to compare the traditional CNN results with those using K-nearest Neighbors(KNNs) and Support Vector Machines(SVMs). However, within a week of testing, it became evident that KNNs and SVMs would not be viable models to use for this experiment. To ensure that our data was uniform, we had to resize each image to be 300 by 300 pixels and store channels for each of the three RGB values. As these two models were linear models, the ≈200,000 dimensional flattened vector that resulted for each image was far too large. As such, to ensure enough detail was captured in our models, we pivoted to using the latest CNNs.

Additionally, we wanted to gauge the impact that identifying objects would have on the overall outcome of indoor scene recognition. Identifying objects within an indoor scene can be incredibly beneficial in accurately determining the environment. These objects frequently provide context and act as important identifiers for the scene. For instance, a dining room would usually have a table, a few chairs, and some cutlery, like plates, forks, spoons, and knives. While these objects may appear in other images as well, it is incredibly unlikely that they all appear in the same image and that image is not of a dining room. Thus, more accurate scene classification may result from the recognition of these objects. Moreover, objects can offer insightful context clues that supplement other visual characteristics and raise the classification process's overall accuracy. On the other hand, there are potential drawbacks in regards to identifying objects. Not every object in a scene will always be useful in identifying a particular scene. Additionally, relying solely on object recognition may overlook minute details in the layout and composition of the scene, which could result in problems like overfitting. For example, if from the data provided, the only time mugs are present are in a cafe, an image of a dishware store selling mugs would likely get misclassified. Moreover, classifications may become ambiguous in certain scenes due to common objects found in various types of environments. Furthermore, there simply may not be many identifiable objects in a given scene. For example, it is not uncommon to come across an empty train station. But the presence of a few benches and signs may not be sufficient to identify the scene's location. As such, instead of focusing solely on object recognition, it may be more effective to leverage a combination of objects detects, their relative locations, and contextual information to achieve robust scene classification.

In conclusion, our project aims to explore the efficacy of the latest CNNs in contrast to traditional CNNs for indoor scene identification. By evaluating their performance side by side, we hope to gain insights into the advancements made in the field of computer vision. Additionally, we aim to explore the potential benefits of incorporating object detection techniques into the scene identification process. While object recognition may offer valuable contextual cues, our study will critically assess its contribution alongside other visual features and spatial relationships. Through this comprehensive analysis, we aspire to advance our understanding of scene classification methodologies.

2 RELATED WORK

With the rising interest in computer vision, scene detection is a subfield that is being worked on with several different models and tools being built to aid in this problem. With this however, the current research on scene detection that exists now frequently uses either outdated CNN architectures or CNN architectures that require an unfeasibly large amount of computational power. Furthermore, advancements in ResNet and Inception CNNs pose the potential to make new headway in this area. The following are a few instances of recent research addressing this issue:

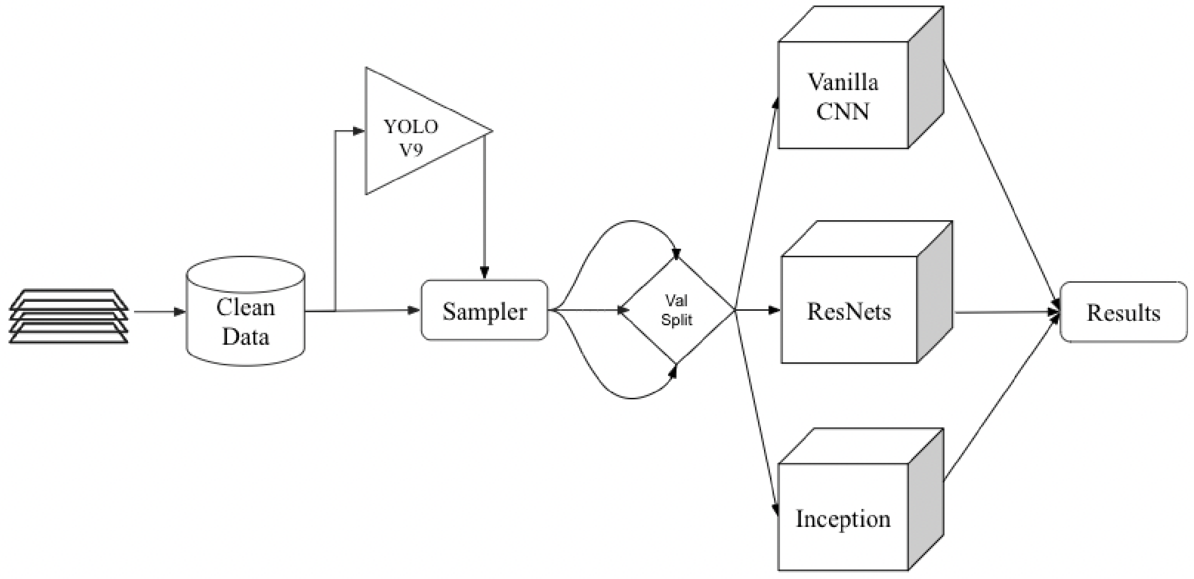

Figure 1: Image Pipeline

Seong et al. [2] introduce a new approach to scene recognition called FOSNet, Fusion of Object and Scene Net. FOSNet unifies this data via a trainable fusion module known as correlative context gating, or CCG for short, which is uncommon in conventional techniques. This module captures the relationship between objects and their surrounding environment, which in turn enables a more thorough understanding of the scene. A new loss function called scene coherence loss, or SCL, was created specifically for scene recognition tasks in order to train FOSNet. SCL makes use of the inherent qualities of scenes, like the way that scene elements are distributed spatially and how scene classes remain constant throughout an image. SCL makes learning and representation of scene features more efficient by incorporating these special qualities. According to the experimental evaluations, FOSNet performed exceptionally well on the MIT Indoor 67 datasets.

Dosovitskiy et al. [3] explored an innovative way of using Transformers directly for image recognition tasks. The method differs from previous methods that included image-specific biases in their architectures, with the exception of the first patch extraction stage. Instead, they handle an image as a collection of patches and process it using a standard Transformer encoder, which is akin to those used in natural language processing, or NLP. The outcomes of this straightforward yet scalable method are remarkable, particularly when combined with pre-training on sizable datasets. They achieved performance levels that either match or surpass the state of the art on a number of image classification datasets using this method called the Vision Transformer. Notably, this accomplishment requires comparatively little pre-training, which makes it a desirable choice for image recognition practitioners looking for economical yet reliable solutions.

Lopez-Cifuentes et al. [4] presented an innovative approach to scene recognition utilizing an end-to-end multi-modal CNN architecture. Unlike conventional approaches, the method captures both image and contextual information through a dual-branched CNN. This includes the traditional RGB branch alongside a complementary semantic information branch. The features from these branches are combined via an attention mechanism. Out of all of these, one particularly unique strategy is one involving a softmax transformation applied to the convolved Semantic Branch features. These transformed features serve to modulate the convolved features of the RGB Branch, effectively enhancing the network's ability to learn pertinent contextual cues. By directing attention towards human-interpretable concepts indicative of scene classes, this method reinforces the acquisition of relevant contextual information. The resulting method outperformed every method in the MIT Indoor 67 set of methods at the time while reducing the number of parameters.

3 EXPERIMENT

With the main focus of the study being exploiting local and global discriminative information, our focus included scene classification with and without object identification. However, before doing this, it was imperative that the images passed in were standardized in terms of size and shape to ensure that the results remained unbiased. After this, to identify objects within the images, images were passed into a pre-trained semantic segmentation model called YOLOv9 (You Only Look Once). While utilizing object detection models may have provided some results, semantic segmentation provides a comprehensive understanding of the scene by segmenting all objects present in the image, including background elements, which we determined to be far more effective in scene classification. To add on, we incorporated YOLOv9 into our pipeline as it has real-time object detection, offering significant improvements in terms of efficiency, accuracy, and adaptability [6]. With the local discriminative information provided by YOLOv9, the question became how to

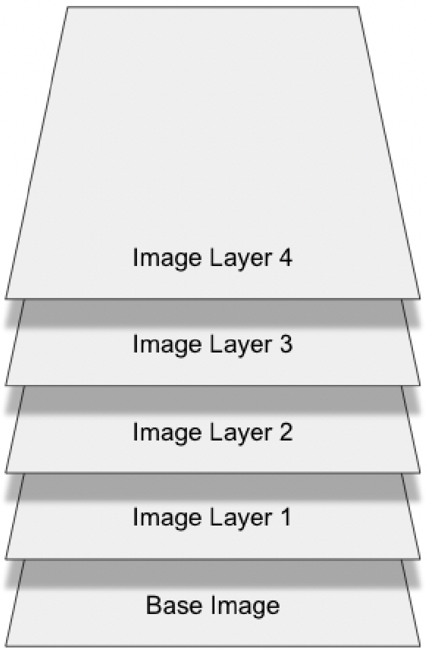

Figure 2: Stacking Bounding Boxes Masks

leverage it since the output was given in layers in terms of the local information. There were three main options that were considered:

- Adding the values provided for each layer up onto the original image.
- Stacking these different layers on top of one another to create a tensor for the image as shown in Figure 2.
- Incorporate ideas from the transformer architecture and learn positional encodings for the objects.

Fortunately, all of these perform better than just inputting the image itself into either of the pre-trained models. Following this, our data underwent sampling procedure to facilitate model training. Again, there were three distinct sampling techniques in contention:

- Simple random sampling: Each sample has an equal probability of being chosen.
- Stratified sampling: Samples specific proportions of individuals from various subpopulations in the larger population.
- Sequential sampling: The order determined by the split created by the authors of the dataset.

Ultimately, after conducting several tests, we found that the sampling technique that performed the best was stratified sampling with results being 2-3% better on average. However, it came at a serious cost as the time taken to run stratified sampling was significantly more than either of the other two techniques. From here, data was passed into the three models as shown in Figure 1.

4 DATASET

In this project, we used the MIT Indoor 67 dataset [1]. This is a database consisting of 15,620 images divided into 67 different indoor scene categories. While the number of images differs from scene to scene, each scene has at least 100 images assigned to it, all in JPG formats. The total size of this database is 2.4 gigabytes and includes images with varying characteristics, like size, image quality, and lighting.

In order to ensure that all images were read in the same way, we resized all of the image such that they were all 300 pixels by 300 pixels. The images were resized using torchvision.transform where they were first horizontally resized and then cropped to get the desired size.

5 METHODOLOGY

5.1 K-Nearest Neighbors

K-Nearest Neighbors (KNN) [11] is a simple, versatile and widely used machine learning algorithm. It is a type of instance-based learning, where the function is only approximated locally and all computation is deferred until classification. The KNN algorithm operates on a very simple principle: it classifies a data point based on how its neighbors are classified. The algorithm works by computing the distance between each query instance and all the examples in the training dataset. It then sorts all the distances and chooses the ‘k’ nearest neighbors. The new point is then assigned the class most common among its ‘k’ nearest neighbors.

5.2 Support Vector Machines

Support Vector Machines (SVMs) [12], a set of supervised learning methods used for classification problems, work by contructing a hyperplane or set of hyperplanes in a high-dimensional space that optimally separate the dataset into classes. SVM performs linear classification, which means finding the hyperplane that separates the classes with the maximum margin. The primary goal in training an SVM is to find the coefficients of the hyperplane equation. This is done by solving a convex quadratic optimization problem that seeks to maximize the margin between the classes while minimizing classification errors. Critical elements of the dataset, known as support vectors, are the data points nearest to the hyperplane. The decision function of the SVM is influenced only by these data points because they are the most difficult to classify.

To use KNNs and SVMs with our dataset, We first flattened our 3 dimensional (RGB) images into one dimensional vectors. Then, we applied Principal Component Analysis (PCA) to reduce the number of features to [50, 100, 150]. We used the resulting vector to train the SVM/ KNN classifiers.

5.3 Convolutional Neural Networks

Convolutional Neural Network (or CNN) is a regularized type of feed-forward neural network that learns feature engineering by itself via filters (or kernel) optimization. CNNs are particularly effective for processing grid-like data, such as images due to their ability to capture spatial hierarchies of features in visual data. Our CNN consists of several convolutional layers, each followed by a ReLU activation function and a max pooling layer. This sequence is designed to extract and reduce feature dimensions progressively while retaining the most significant features.

5.3.1 Convolutional Layers. Each convolutional layer applies several learnable filters to the input image to create feature maps. Each filter detects spatial hierarchies of features ranging from simple edges in the initial layers to complex features (like textures and patterns) in deeper layers. The convolution operation helps the network focus on high-importance areas by enhancing features that are crucial for analysis. The operation can be expressed as:

$$A_{ij} = \sum_{m} \sum_{n} F_{mn} X_{i+m,j+n}$$

Where F is the filter matrix and X is the output matrix.

5.3.2 ReLU Activation Function. This layer applies the non-linear ReLU function

$$f(x) = max(0, x)$$

to each pixel of the feature map. The ReLU activation function is used to add non-linearity to the model, allowing it to learn more complex patterns. It helps in speeding up the training process by overcoming the vanishing gradient problem commonly seen with sigmoid or tanh functions.

5.3.3 Max Pooling Layers. Following convolution and ReLU layers, max pooling is used to reduce the dimensionality of each feature map. It works by sliding a 2x2 window across the feature map and keeping only the maximum value from each window section. This is represented by the following max filter:

$$P_{ij} = max_{a,b \in W} X_{i+a,j+b}$$

This process helps in making the representation smaller and more manageable, and it provides an abstracted form of the features being detected in the input.

5.3.4 Fully Connected Layers. After several stages of convolution and pooling, the high-level reasoning in the network is done via fully connected layers. Neurons in a fully connected layer have connections to all activations in the previous layer. These layers essentially learn non-linear combinations of the high-level features as represented by the output of the convolutional/pooling layers. The final fully connected layer (combined with a softmax activation) produces the distribution of class probabilities for the input image.

5.4 ResNet-18

ResNet-18 [9] is a specific configuration of the Residual Network (ResNet) architecture, a type of Convolutional Neural Network that was designed to enable the training of much deeper networks. The primary innovation of ResNet is its use of residual blocks with shortcut connections, which help to address the problem of vanishing gradients—a challenge that arises in traditional deep neural networks as depth increases. ResNet-18 consists of 18 layers that have weights; this includes convolutional layers and fully connected layers but excludes pooling layers and batch normalization layers, which do not have learnable parameters.

5.4.1 Initial Convolution and Pooling. The network begins with a single 7x7 convolutional layer with 64 filters, followed by a batch normalization layer, a ReLU activation, and a max pooling layer. This setup prepares the input for the subsequent residual blocks.

5.4.2 Residual Blocks. The core idea of ResNet is introducing a shortcut connection that skips one or more layers. Traditional neural networks try to directly learn the underlying mapping for data, while ResNet learns the residual or difference between the input and output of a stacked layer. This is formulated as

$$\text{Output} = F(\text{input}) + \text{input}$$

where $F(x)$ represents the mapping learned by the stacked layers and x is the input to the block of layers. By allowing these shortcuts, ResNet allows training of much deeper networks by easing the flow of gradients during the backward pass in training. The core of ResNet-18 consists of eight residual blocks. Each block has two convolutional layers with 3x3 filters. Residual blocks in ResNet-18 are structured into four groups, each with a different number of filters (64, 128, 256, and 512) and each group containing two blocks. The blocks use identity shortcuts when the input and output dimensions match. When the number of filters increases, the stride changes to 2 in the first convolutional layer of the block to reduce the spatial dimensions, and a 1x1 convolutional shortcut is used to match dimensions.

5.4.3 Custom Modifications. The original fully connected layer of the ResNet-18 model is replaced with a new linear layer. ResNet-18's final convolutional layer outputs 512 features, which are input to this layer. The replacement linear layer transforms these 512 features into a number of outputs equal to the number of scene categories. This modification tailors the model's output to the specific classification task.

5.5 ResNet-34

ResNet-34[9] is another configuration of the Residual Network (ResNet) family, similar to ResNet-18 but with increased depth. It's designed to handle even more complex datasets and tasks, offering improvements in learning capability due to its deeper structure. ResNet-34, like ResNet-18, uses residual blocks with shortcut connections to facilitate the flow of gradients through the network, allowing for effective training of deeper networks. The primary difference lies in the number of layers.

5.5.1 Total Layers. ResNet-34 comprises 34 layers that have weights, including convolutional and fully connected layers. Like ResNet-18, pooling and batch normalization layers are also used but do not count towards the total layer count as they do not have learnable parameters.

5.5.2 Initial Setup. ResNet-34 starts with the same setup as ResNet-18—a 7x7 convolutional layer with 64 filters followed by batch normalization, a Sigmoid activation, and a max pooling layer.

5.5.3 Residual Blocks. The main distinction is in the number and arrangement of the residual blocks. ResNet-34 uses sixteen residual blocks organized into four groups by the number of filters (64, 128, 256, 512). Each group contains more blocks than in ResNet-18:

- 3 blocks with 64 filters each
- 4 blocks with 128 filters each
- 6 blocks with 256 filters each
- 3 blocks with 512 filters each

Each residual block consists of two 3x3 convolutional layers. The identity shortcut is used when the dimensions match, and a 1x1 convolutional shortcut is employed for dimension matching when the number of filters increases, typically accompanied by a stride of 2 in the first convolutional layer of these blocks to reduce spatial dimensions.

5.6 Inception Network

The Inception Network[10] represents an advanced CNN architecture that builds on the idea of how am optimal local sparse structure of a convoluted visual network can be approximated and covered by readily available dense components. These networks incorporate a complex arrangement of convolutional layers and operations that allow them to capture information at various scales simultaneously, making them highly effective for detailed and nuanced image analysis. The key feature of Inception networks is their use of "Inception modules". These modules consist of several parallel paths that include convolutions of different sizes (1x1, 3x3, 5x5) and pooling operations. This design allows the network to capture and process spatial hierarchies in images at multiple scales, from very local features (through smaller convolutions) to more global features (through larger convolutions and pooling). This multi-scale processing is particularly advantageous for complex image tasks where different objects and features can vary significantly in size and context.

We replace the original fully connected layer of inception-v3 with a new linear layer that maps the 2048 feature outputs from the last pooling layer of inception-v3 to our number of classes. This modification adapts the network to output probabilities for the number of scene categories relevant to our problem.

6 EXPERIMENTAL SETUP

6.1 YOLO-v9

We utilize YOLO-v9, a state-of-the-art deep learning model for real-time object detection, to identify and localize objects within various indoor scenes. We load a pre-trained YOLO-v9 model which is capable of detecing multiple object classes and their locations within an image. For each detected bounding box, a new image layer is created to mark the region corresponding to each object class as seen in Figure 3.

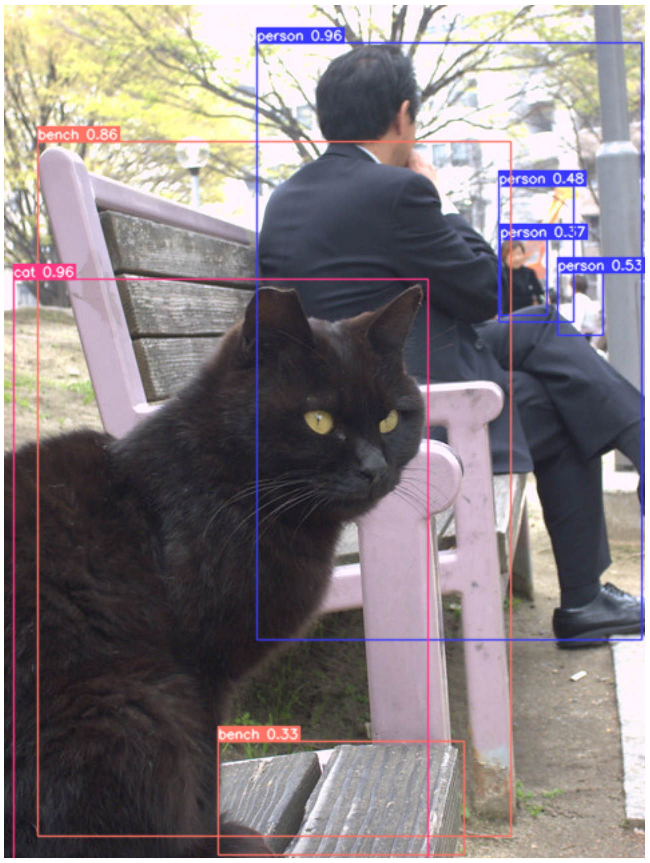

Figure 3: YOLO-v9 Output from [7]

6.2 Models

For each of the models (CNN, ResNet, and Inception Network), we extend PyTorch's `nn.Module` to create a custom neural network module. We load pre-trained models with default weights as a starting point. In the case of ResNet and Inception Network, this gives us the additional benefit of having models trained on the ImageNet dataset [8], which is a large dataset of over a million images categorized into 1000 different classes. This helps the model learn and interpret a wide range of visual features. We then fine-tune the pre-trained model by adapting the final layer and writing a custom forward pass.

We replace the fully connected final layer of the original models with a custom linear layer which uses a linear transformation from the 512 features generated by the preceding convolutional layers to the number of classes in our dataset (67). This modification tailors the network's output to the specific dataset and problem being addressed. In the forward pass, we apply the sigmoid activation function to the output of the model for each model. The sigmoid function is used to map the logits from the linear layer to probabilities between 0 and 1. This allows us to predict each class independently (as is typical in multi-label classification) and enables the model to output a probability score indicating the presence of each class.

6.3 Hyperparameter Tuning

After tuning our hyperparameters, we found that are models performed best with the following hyperparameters:

- Learning rate: 1e-5
- Batch size: 64
- Optimizer: AdamW
- Loss function: cross-entropy loss

6.4 Training and Validation

Model training and evaluation are managed using custom data loaders that handle image input and batching operations. We employ the AdamW optimizer with a learning rate of 1e-5 for training out models. The models are trained over 20 epochs with a batch size of 64.

The model is evaluated using a K-Fold cross-validation approach, ensuring robustness and generalizability of the model across different subsets of data as well as unseen data. Each fold splits the dataset into a training set and a validation set, with different partitions used in each iteration to ensure comprehensive evaluation. Losses and accuracies are recorded for each fold, and the model parameters are optimized using the Adam optimizer.

The primary loss function used for optimization is the cross-entropy loss, suitable for multi-class classification tasks. The loss is defined as:

$$L = -\sum_{c=1}^{M} y_{o,c} log(p_{o,c})$$

Where M is the number of classes, y indicates whether class c is the correct classification for observation o, and p is the predicted probability of observation o being of class c.

Each epoch's training and validation losses and accuracies are calculated and printed to monitor progress and performance.

6.5 Evaluation Metrics

After training, the model's performance is analyzed on unseen test data, providing insights into the generalizability and effectiveness of the trained models. Model performance is assessed using accuracy, precision, recall, and F1-score, calculated for both training and validation phases. Additionally, accuracy on the test set provides a measure of the model's ability to generalize.

7 RESULTS AND DISCUSSION

7.1 Determining the Value of Bounding Boxes

For each model that we implemented, we tested the accuracy and F1 scores of the model both with and without bounding boxes. With each 3 trials run per model, the results we reported are in the tables below. With ResNet performing the best, specifically ResNet34 with Bounding Boxes, compared to all of the other models. Because of this, we proceeded to run our tests with ResNet34.

7.2 KNNs/ SVMs

From the results in Table 1, we can see that KNNs/ SVMs perform quite poorly both with, and without bounding boxes, with an accuracy of only 8.23%. We were originally hoping to use these models as a benchmark for the CNNs, but after seeing these results, that was no longer feasible.

We conclude that this inadequacy can be attributed primarily to several key issues associated with the nature of the image data and the inherent limitations of these models.

Firstly, KNNs and SVMs struggle with the high dimensionality typical of image data. Known as the "curse of dimensionality," this issue makes the data sparse in such a large feature space, complicating accurate distance measurement (critical for KNNs) and effective hyperplane placement (used in SVMs). Specifically, KNN suffers because distance metrics become less meaningful in high-dimensional spaces, whereas SVM, although capable of handling higher dimensions using the kernel trick, requires significant computational resources and careful kernel selection, which can be complex and not always feasible.

Moreover, the pre-processing step of flattening images into one-dimensional vectors results in substantial loss of spatial and contextual information. Such flattening eliminates crucial details about

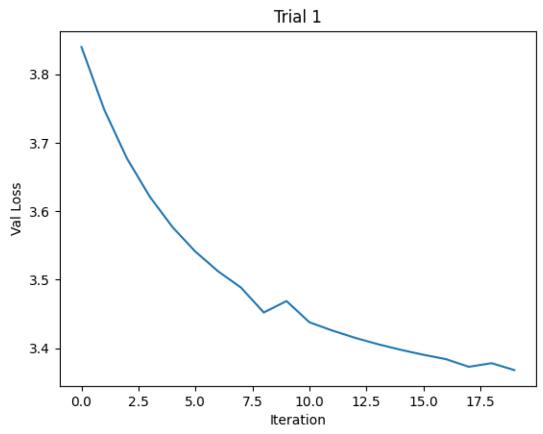

Figure 4: ResNet34 Trial 1 Validation Loss

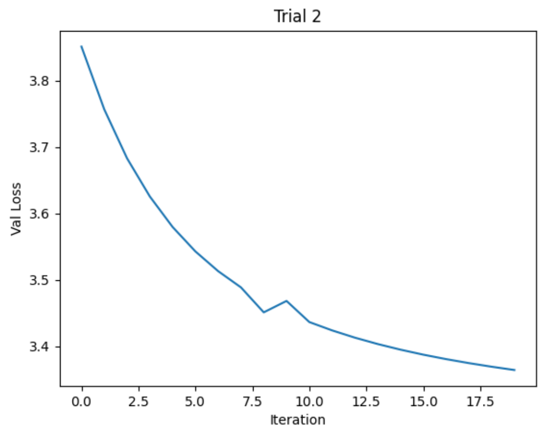

Figure 5: ResNet34 Trial 2 Validation Loss

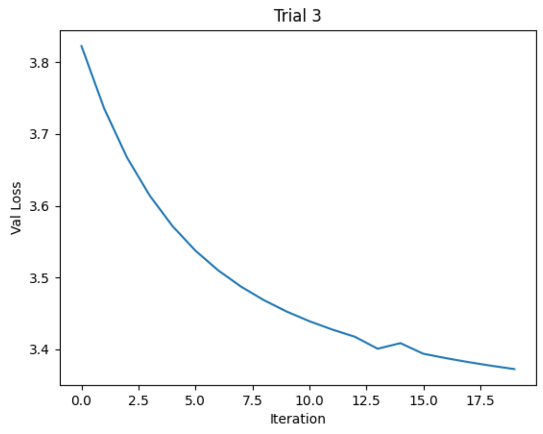

Figure 6: ResNet34 Trial 3 Validation Loss

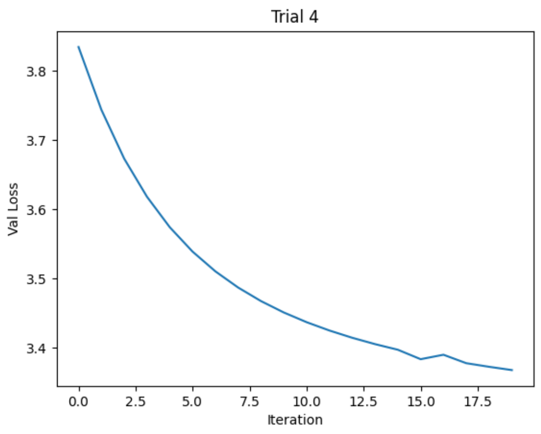

Figure 7: ResNet34 Trial 4 Validation Loss

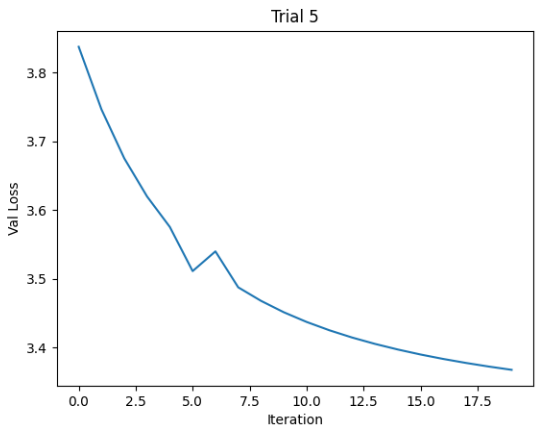

Figure 8: ResNet34 Trial 5 Validation Loss

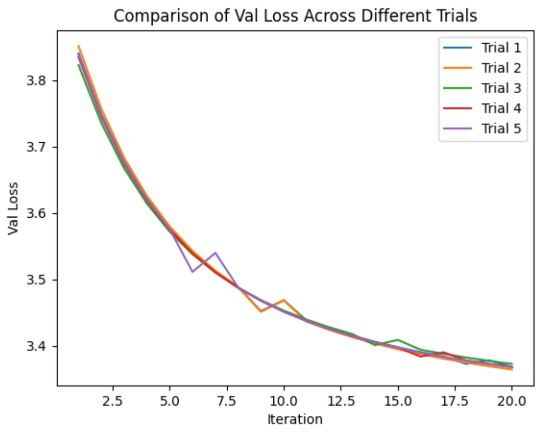

Figure 9: ResNet34 Validation Loss of All Trials

Trial Number	ACC with BB	F1 with BB	ACC without BB	F1 without BB
1	7.31	6.26	6.84	5.11
2	**8.23**	7.73	**7.57**	4.58
3	6.24	6.93	5.95	3.84

Table 1: Performance of KNN and SVM With and Without Bounding Boxes

Trial Number	ACC with BB	F1 with BB	ACC without BB	F1 without BB
1	20.45	14.43	18.32	13.4
2	**20.83**	15.22	20.11	14.32
3	19.32	13.53	**21.42**	14.67

Table 2: Performance of Vanilla CNN With and Without Bounding Boxes

Trial Number	ACC with BB	F1 with BB	ACC without BB	F1 without BB
1	75.86	75.51	**73.23**	73.12
2	**76.81**	76.43	72.85	71.29
3	75.95	75.67	73.12	72.54

Table 3: Performance of ResNet18 With and Without Bounding Boxes

Trial Number	ACC with BB	F1 with BB	ACC without BB	F1 without BB
1	76.51	76.67	**73.58**	72.23
2	**77.65**	78.12	72.11	71.80
3	76.87	76.66	72.65	72.32

Table 4: Performance of ResNet34 With and Without Bounding Boxes

Trial Number	ACC with BB	F1 with BB	ACC without BB	F1 without BB
1	74.28	74.03	**73.27**	73.09
2	74.72	74.54	73.08	72.98
3	**75.71**	75.41	73.25	72.86

Table 5: Performance of Inception With and Without Bounding Boxes

Model	Best ACC	Average ACC	Best F1	Average F1
KNN and SVM	8.23	7.26	7.73	6.93
Vanilla CNN	21.42	19.95	15.22	14.39
ResNet18	76.81	76.20	76.43	75.87
ResNet34	**77.65**	77.01	**78.12**	77.15
Inception	75.71	74.90	74.72	74.66

Table 6: Best and Average Performance of All Models

the structural relationships between pixels, which are vital for recognizing and differentiating between various indoor scenes. Consequently, both KNN and SVM fail to capture the complex, non-linear

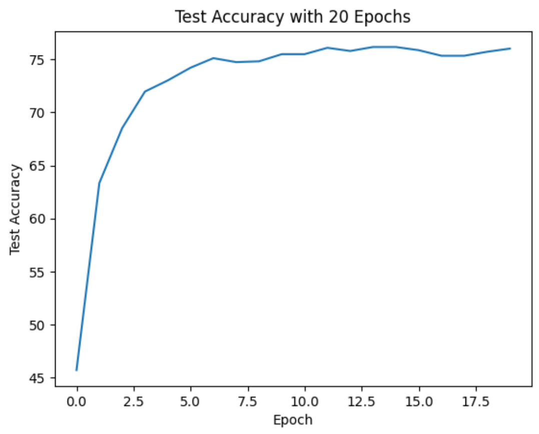

Figure 10: Best ResNet34 Test Accuracy Across 20 Epochs

patterns present in the image data, further hampering their effectiveness. These challenges suggest a need to explore alternative models like CNNs, which inherently account for spatial hierarchies and can more adeptly handle the intricacies of image data without the significant pre-processing drawbacks seen in KNN and SVM.

7.3 CNNs

From the results in Table 6, we can see that the ResNet model yields the best results. Despite having a similar parameter count as the more sophisticated ResNet and Inception models, performed the worst in our experiments. This outcome can primarily be attributed to the architectural simplicity and lack of advanced mechanisms like skip connections or multiple filter sizes per layer, like in the ResNet and Inception architectures respectively.

Vanilla CNNs, which consist of straightforward convolutional layers stacked one after the other, lack mechanisms to effectively preserve information across deeper layers of the network. Without features like skip connections, these models are more prone to the vanishing gradient problem, where gradients diminish as they backpropagate through each layer. This issue severely affects the training process, particularly in deeper networks, leading to poorer generalization capabilities on limited datasets. As a result, despite having a comparable number of parameters, vanilla CNNs do not leverage them as effectively and lead to inferior performance.

7.4 ResNet

Our hypothesis was that models often under-perform in these scenarios because they do not have enough data to learn effectively, often resulting in convergence to local minima and they acquire of non-generalizable and "silly" features.

For ResNets pre-trained on ImageNet, which contains a vast variety of images, these models bring a broader understanding of image features. This prevents them from focusing on irrelevant details and helps in achieving more robust performance.

Mathematically, when training models on small datasets, the goal is to find the optimal parameters (θ^*) by minimizing the loss function (L):

$$\theta^* = \arg\min_{\theta} L(\theta)$$

However, this process often results in convergence to local minima. Pre-trained models assist by providing a better initial point ($\theta_{\text{pre-trained}}$), and the training process is then adjusted to:

$$\theta^* = \arg\min_{\theta} L(\theta) + \lambda \|\theta - \theta_{\text{pre-trained}}\|^2$$

Here, λ is a regularization term that keeps the learning close to what was learned from the large dataset. Hence, ResNets performed better than vanilla CNNs.

7.5 Inception

Inception model performed worse than the ResNet in our experiments due to several architectural differences affecting their training and generalization capabilities on limited datasets. The Inception model, with its complex structure featuring multiple convolutional filters of varying sizes at each layer, then concatenating the outputs, requires a more substantial amount of data to effectively generalize. This complexity can lead to overfitting on smaller datasets, where the model might learn noise instead of useful, abstract features.

Furthermore, the optimization and initialization of such a complex network are more challenging, potentially leading to less stable training outcomes. In contrast, the ResNet architecture benefits from its simpler, repetitive blocks equipped with residual connections. These connections help prevent the vanishing gradient problem, allowing for deeper networks that maintain effective learning capabilities across additional layers, thus ensuring more robust training and better performance on tasks with limited data availability.

7.6 Comparing the Models

From the results in Tables 1 through 5, we can see that with the exception of the Vanilla CNN, all models performed better with the incorporation of bounding boxes. This indicates that, for the most part, the inclusion of bounding boxes is beneficial to indoor scene classification. Now, in order to determine which model runs best, we can take their maximum scores as well as their average scores, as shown in Table 6. In this table, we can make three key observations. First, we can see the lack of accuracy and F1 score in the KNN and SVM models. As explained before, this was something that we had expected given how the information would translate and be compressed to this model. Second, we can see that the Vanilla CNN performed better than the KNN and SVM, but still, it was considerably lower than the three models. This only emphasizes the fact that the other CNN models have significantly more depth. Third, we can see that while the rest of the models have relatively similar scores for both accuracy and F1, ResNet34 is the model that has the best average and best overall scores for both accuracy and F1. This result underscores the effectiveness of ResNet34's architecture in capturing intricate features and patterns inherent in scene images. The performance of ResNet34 suggests its suitability for scene classification tasks, offering reliable and accurate predictions across a diverse range of scenes. Moving forward, we used ResNet34 as the primary model architecture.

7.7 Results Using ResNet34

With ResNet34 with Bounding Boxes performing the best, we ran 5 trials consisting of 20 epochs in each to gauge the validation loss. In Figures 4 through 8, we can see that they all begin with relatively high validation losses but all decrease at similar rates as the epoch iteration increments. In Figure 9, we can see the similarity between all of the values between each trial. This trend indicates that the ResNet34 model with Bounding Boxes consistently improves its ability to generalize unseen data over the course of testing. The consistent reduction in validation loss suggests that the model is effectively learning from the training data and capturing relevant patterns in scene classification without overfitting. These results provide confidence in the reliability and generalization performance if the model, reinforcing its ability to accurately classify scenes.

Additionally, using Figure 10, which highlights the best test accuracies of the 20 epochs running the ResNet34 model, we can see that test accuracy quickly jumps within the first few epochs only to plateau around 77%. This indicates that the model is able to quickly adapt and learn, however, the fact that it plateaus around the 15^{th} epoch indicates that testing for more epochs will see minimal improvement. Across these tests, however, we saw that the highest test accuracy occurred at epochs 14 and 15 with a test accuracy of 77.65%.

8 CONCLUSION

In this report, we investigated the impact of object definition using bounding boxes on indoor scenery classification using advanced Convolutional Neural Networks. We compared traditional methods, various CNN architectures, and transfer learning approaches. Our results revealed that incorporating bounding boxes improved accuracy by 3-4% in ResNet models, but had no significant effect on the Inception model. Both architectures outperformed traditional CNNs, likely due to their complex structures and fine-tuning on ImageNet.

Despite these gains, the additional input information introduced by bounding boxes occasionally reduced performance, suggesting a need for more sophisticated object representation methods like positional encodings or a more sophisticated model like Vision Transformer [3] or FOSNet [2]. Hence, while object-focused features can enhance CNN performance, the balance between feature sparseness and model efficiency is important.

REFERENCES

[1] Quattoni, A., & Torralba, A. (2009). *Recognizing Indoor Scenes.*

[2] Seong, H., Hyun, J., & Kim, E. (2019). *FOSNet: An End-to-End Trainable Deep Neural Network for Scene Recognition.*

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.*

[4] Lopez-Cifuentes, A., Escudero-Vinolo, M., Bescos, J., & Garcia-Martin, A. (2019). *Semantic-Aware Scene Recognition.*

[5] Song, C., & Ma, X. (2023). *SRRM: Semantic Region Relation Model for Indoor Scene Recognition.*

[6] Wang, C., Yeh, I., & Liao, H. (2024). *YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information*

[7] Convert and optimize yolov9 with openvinoTM. OpenVINO. (n.d.). https://docs.openvino.ai/2024/notebooks/287-yolov9-optimization-with-output.html

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90

[10] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 2818-2826, doi: 10.1109/CVPR.2016.308

[11] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, pp. 1883, 2009.

[12] E. Mayoraz and E. Alpaydin, "Support vector machines for multi-class classification," in *Engineering Applications of Bio-Inspired Artificial Neural Networks*, eds. J. Mira and J.V. Sánchez-Andrés, vol. 1607 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, 1999, pp. [Page Numbers], doi: 10.1007/BFb0100551.