university of groningen / faculty of science and engineering

Master Thesis (CS) Project Report
Topic Modelling for Place Classification using Deep Learning
WMCS901-30

Rishabh Sawhney (S4133366)

First Supervisor : Estefania Talavera Martinez(University of Groningen)
Second Supervisor : Michael Biehl (University of Groningen)

October 12, 2021

Abstract

Deep Learning Models have been used extensively for Place Classification. The aim of this project is to check whether the probabilistic approaches such as LDA can be used in Deep Learning models in order to perform Place Classification. Topic modelling is chosen as a technique, aiming to automatically derive topics during the training of Deep Networks for the task of place recognition. The Datasets used for the project are EgoPlaces (A modification of the EgoPlaces dataset) as well as the Places365 Dataset. The results of the Topic model architectures proposed are compared with the results from a baseline model which is used for Place Classification. The topic models were able to reach around 70% testing accuracy for the EgoPlaces dataset as well as around 10% accuracy for the Places365 dataset with around 12% top-1 prediction accuracy for the EgoPlaces dataset. The Topic models are not able to achieve comparable results to the baseline Place Classification model. However, it does exhibit the distribution of 1000 objects in into different Topics for a particular place for 4 and 9 topics.

Contents

1 Introduction

1.1 Problem

In recent years, with the help of advancement in computation capabilities of machines, Deep Learning has been the norm and is actively being pursued across various industries ranging from Pharmaceuticals, Bio-informatics to day to day applications in social media analysis. Deep Learning networks allow humans to help machines develop semantic understanding of the type of data being processed which allows it to analyse and decipher patterns in the data and process it in different layers through the use of Convolutions. These patterns stored in the intermediate layers of the model are then passed into the subsequent output which is typically a fully connected layer which provides the probability of the input belonging to a particular class and thus help in predicting the labels of the data. The data can range from Time Series, Images to even speech through the corresponding data pre-processing techniques. Thus all the factors listed above has enabled us to build upon the existing Machine Learning frameworks as seen in Figure 1. Through adding the Convolution Layers in the network, one can exercise greater control over the training as well as the prediction process for Classification as well as Regression Problems for different types of data. Topic

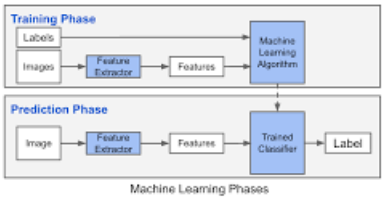


Figure 1: Traditional Machine Learning Approach[28]

Modelling is a probabilistic tool that is being extensively used in the semantic analysis of text and is used in fields such as text mining and semantic analysis to name a few. It makes use of Latent Dirichlet Allocation(LDA) which creates sparse probabilities priors which are encoded over Topic - Document and Topic - Word probabilities. This methodology is based on the pretense that a topic encases a sparse set of Documents and the topics utilize a small amount of words in the document[8]. LDA is usually processed in the form of a Document-Term matrix which comprises of a list of documents as rows and the number of occurrence of each word in each document as the columns. This will be better explained in the sections given below.

Even though the application of LDA based Topic Modelling is widespread in the text analysis domain, its applications are few and far between when it comes to alternate forms of data such as images. This makes us arrive at the following question :

'Can Deep Learning models be utilized in order to perform such Probabilistic based approaches in some way?.'

'And if so, how do these fare when compared to Multi-Class Image Classification Deep Networks on the same image data?'

1.2 Aim

The aim of this project attempts to answer the above question through the means of devising network architectures which would perform an algorithm similar to LDA in order to organize the Images labelled into different class probabilities into topics which are automatically assigned by the Deep Network. The number of topics is specified by the user and contains set of image classes belonging to it. Thus in a broader sense, through this model, one should be able to map places to topic probabilities from an image Dataset.

1.3 Methodology

The underlying structure of the proposed model is explained in this section. The different model architectures are build upon a pretrained ResNet-50 Network which is trained with the ImageNet dataset. This model is called the backbone model for all the topic model architectures. The output of this network is a softmax dense layer with 1000 units. This softmax layer is connected to the novel Topic Layer which is responsible for creating designated topic weights. These layers are then connected to a Dense softmax layer with a number of units. These units correspond to the number of place categories in the image dataset . The performance of this model is then compared to baseline deep networks fit with the datasets in question and the results for the same are recorded. In the proposed architecture, Dropout layers are implemented in order to prevent the network from over-fitting. An overview of the working of the proposed Topic model can be found in Figure 2.

The softmax output from Resnet-50 yields 1000 values which are used as objects. These objects represent 1000 classes of the ImageNet dataset on which the backbone model is trained on. These topics are then mapped to trainable weights in the novel Topic layer. This could be a layer class which assigns the Topic weights or can be a series of dense layers corresponding to the number of topics , k required. These topic layers performs the classification of the images into the different place categories depending on the number of places in the dataset. This is performed for EgoPlaces as well as the Places365 dataset along with a modification of the Places365 dataset aptly called Places20 dataset for comparison. This is performed for both $k = 4$ and $k = 9$ where k represents the number of topics required.

The methodology will be explained in detail in the later sections of the report.

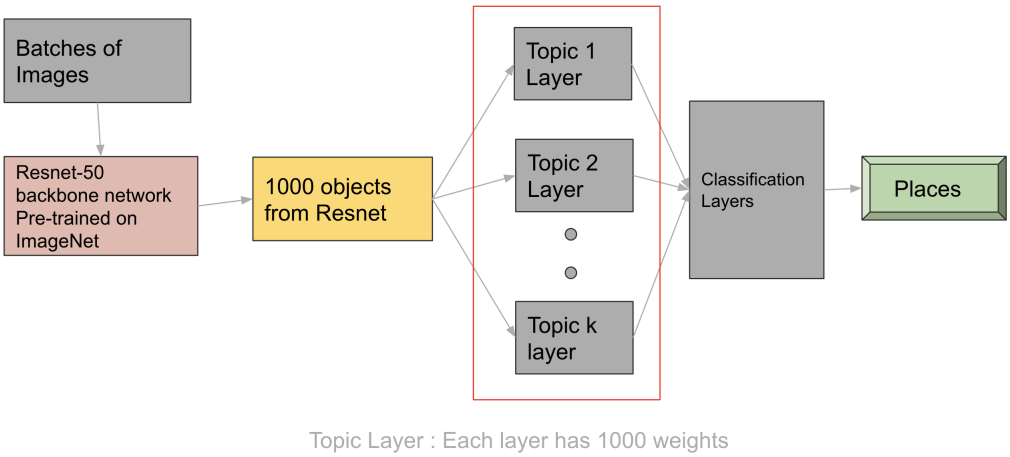


Figure 2: Flow chart showing the working of the proposed Topic Models

1.4 Contributions of this work

This project takes inspiration from Topic Modelling using Latent Dirictlet Allocation(LDA) first proposed by *Blei et. al.*[8] and applying them to Deep Learning Neural Networks. This work focuses on devising different model architectures in order to transform the LDA distributions for k topics through Fully-Connected network layers, one for each topic layer k. The performance of the different architectures of the model are compared against a fine-tuned model which contains the number of different places described in two datasets, EgoRoutine[34] and Places365[39] as the top layer of the model. This particular model in our work is referred to as the Baseline Model. Some Topic Modelling architectures described in the Proposed Methodology Section(Section 5) make use of creating custom Dense network layers which is inspired by the paper *An Ensemble Convolutional Neural Networks for Bearing Fault Diagnosis Using Multi-Sensor Data* proposed by Liu et. al [37]. This custom dense layer will be responsible for evaluating the ensemble methodology created by the k Topics.

All the different model architectures proposed in this work consist of the Resnet-50 model as a backbone for the model with slight modifications which are defined in detail in Proposed Methodology section (Section5). The project also makes use of Transfer Learning first introduced by *Bozinovski et. al*[10]. The backbone model is loaded with weights derived from the ImageNet dataset[13].

The baseline models do not include the top layer from the Resnet-50 model. However, the Topic Model architectures do include the Top layer of the Resnet-50 model which serves as the 1000 objects. The objects are then mapped onto the different places through the subsequent layers of the different Topic Models.

The proposed model architectures along with the creation of the dataset is entirely performed in Python and makes use of TensorFlow and Keras for model creation and evaluation as well as Pandas in order to create and manipulate the

datasets. The functionalities for the different Python libraries introduced above are explained in detail in the Related Works section(Section 3.6).

1.5 Organisation of the report

The report is organized into sections as stated below:

The first section deals with the *Related works* (Section 2) which comprises of the state of the art methodology currently being applied in the space of Place Classification and Topic Modelling. The different methodologies are described in brief along with their working as well as the pitfalls they might have.

The second section is made up of the *Methods used* section (Section 3) where the different terminologies and methods used in the project are explained in detail. This refers to the concepts such as Deep Learning, Topic Modelling, Transfer Learning as well as the different libraries utilized in order to implement the project in Python.

The third section of the report deals with describing the *Datasets* used in the project (Section 4). This section entails the number of images in the different subsets of the different datasets used as well as the number of place categories in each and describing them in detail.

The fourth section comprises of the *Proposed Methodology* (Section 5) which explains the topic model architectures as well as the baseline image classification models used for comparison in detail. This also includes the data pre-processing pipeline strategies required in order to train, evaluate as well as performing place prediction on the different models.

The fifth section includes the *Experiments* (Section 6) performed on the different model architectures proposed. This includes experiments relating to model training and evaluation on the different datasets as well as the defining the validation metrics used for validating the result of the models.

The sixth section of the report summarizes the *Results and Discussion* section obtained for the different topic models proposed along in detail (Section 7). This includes the model prediction results for the different model architectures along with the assessing the topic layers as well as the different topics created by the Topic models which were obtained during the course of the experiments.

The seventh section of the report summarizes the *Conclusions* of the findings from the experiments conducted as well as the results obtained in the sections above (Section 8). This provides high level insights from our analysis in order to answer the question asked in the Introduction section of the project.

The final section of the report provides a framework for future work studies

and some suggested enhancements that can be conducted based on the findings of the project (Section 9).

Additionally, the results for the other model architectures not used in the main section of the report, the topic model architectures with 9 topics as well as some additional information can be found in the *Appendix* section (Section 11) of the report along with the *Acknowledgements* (Section 10) where the contributions for the different authorities who helped in bringing this project to light have been recognized and appreciated.

2 Related Works

This section deals with the state of the art works that have been performed in the domains of Deep Learning and Topic Modelling using probabilistic concepts as well as Image Processing and Computer Vision. These concepts have found wide applications in the fields of audio as well as image recognition.

2.1 Place Classification using semantic segmentation

This concept was first introduced by Yeo et. al.[15] and leverages the concept of Places Classification using semantic segmentation. Semantic segmentation in the case of image processing and computer vision deals with selecting regions of an image and label them according to the annotation data present. This methodology helps in the detection of objects in the image. An example of semantic segmentation can be found in Figure 3. The main idea behind the

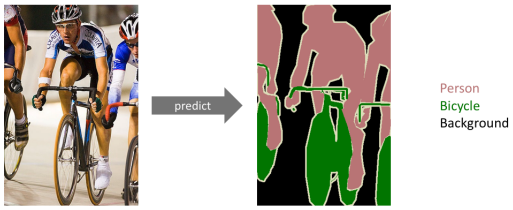


Figure 3: Semantic segmentation used to label portions of an image using a model[15]

work of Yeo et. al. is that humans are better adapted to identify a place or a scene through the background of an image. For example, if the image is belonging to a farm, the first thing the humans notice is that it contains sky thus being an outside place[38]. With this methodology in place, the images are categorized as three main class. These are : indoor, nature and city. All the objects that are identified by the model must belong to either of these classes. The objects in the image are highlighted using semantic segmentation

techniques. The main driving point of this research is the use of a sophisticated weighing matrix scheme proposed by the authors in order to better recognize objects from an image. The methodology for the place classification can be easily understood with the help of Figure 4. An image is first semantically segmented

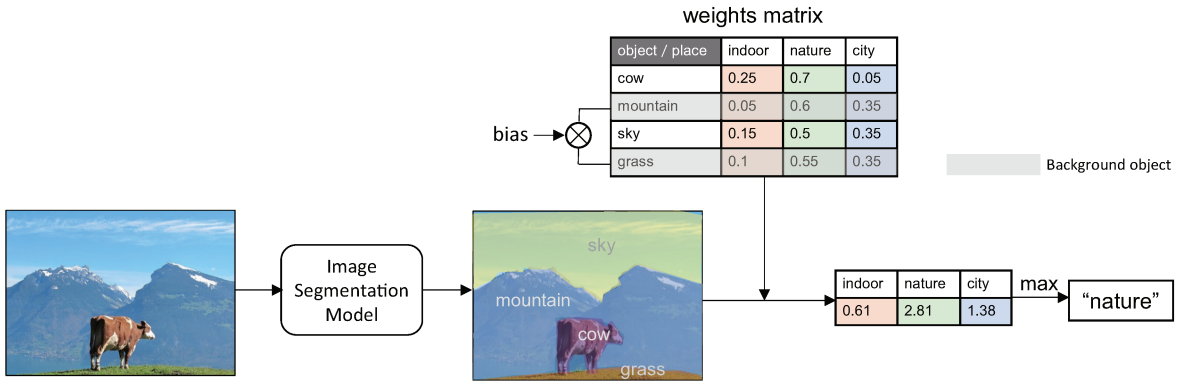


Figure 4: The methodology followed by the algorithm proposed for identifying places from objects in an image[38]

using computer vision techniques through a model. This is followed by applying weights on the different objects in order to render them to one of the three meta classes i.e. indoor, nature and city. The different objects are mapped to these meta-classes. If the model is able to find the objects in the particular image in question, the weights are increased towards its association towards one class. In this work, more emphasis is given towards the detection of background objects identified which are given a higher weight value as compared to the fore-ground objects. Thus this sophisticated approach to place classification is one reason for the model to identify places better than other contemporary place classification models. The weight matrix for the image data has a size of 173x3 which contains 173 objects each having the three scene labels for detecting indoor, nature and city meta-classes[38]. This experiment was performed using the COCO-stuff dataset[21].

This weight matrix forms the main source of learning for the model which is created through using the COCO-stuff meta labels provided which designate the different objects belonging to a certain meta-class. Though the model performs better for recognizing places, it is quite limited by the extent of how exhaustive the weighing matrix scheme will be. Thus the scope of identification of objects will be restricted to a limited scope of the images present in the dataset.

2.2 Assessing data similarity using p-LDA

p-LDA stands for probabilistic Linear Discriminant Analysis. It is a probability based method to assess the similarity between two possibly related samples in the data. This was introduced by Sergey Ioeffe in 2006. This method makes

use of clustering classes by similarity. The advantage of this method is that it is able to gauge similarity in the data at a much higher complexity than other probabilistic techniques used for classification and recognition. This method helps to model a class for an instance in the data using only one example. The objects in the data are considered to be a part of a particular class as a function of their latent variables[19]. LDA is defined as a dimensionality reduction technique used to project the samples of data into a plane of lower dimension such that the distance between similar classes are minimized and the distance between distant classes are maximized in this plane. The prevalence of a class is emboldened through increasing number of samples belonging to the particular class. Class centres in LDA are generated using non-linear functions from the class examples in the data. The p-LDA model assumes that all the samples from the data come from the same data distribution. Classes are designated in this model through the means of a Gaussian Mixture model which associates a sample to a class through the means of its Gaussian. This technique is finding applications in speech recognition as well as to detect facial similarity from previously undocumented faces. A working of the facial identification framework can be visualized using Figure 5.

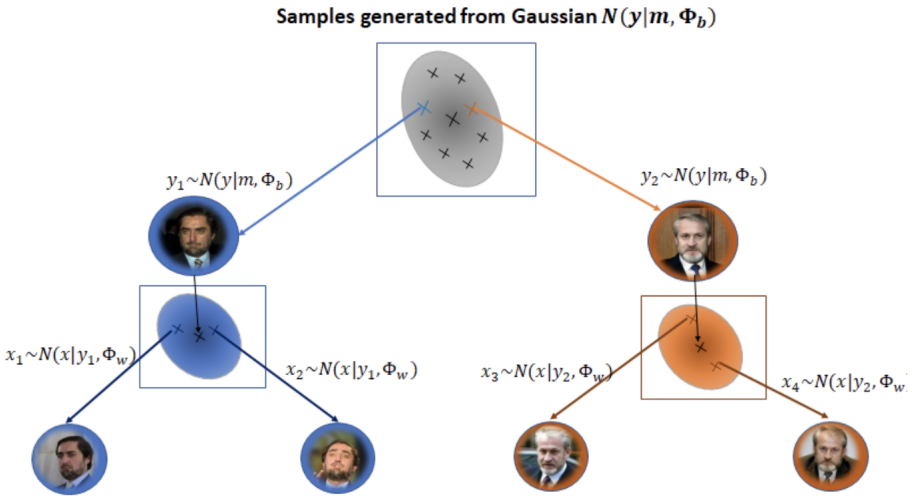


Figure 5: The methodology followed for facial identification using p-LDA[31]

In Figure 5, samples y_1 and y_2 are two classes of faces which are generated from the Gaussian distribution. The model associated the data samples x_1 and x_2 belonging to y_1. Similarly, x_3 and x_4 and assigned to class y_2 by transforming the samples to a latent space. The samples belong to the same face but having different orientation.

The working of pLDA is visualized in Figure 6. Two samples x_1 and x_2 are clustered to the class y in the feature space. This is done by transforming the class centre, y to a value v which is the centre of the class in the latent

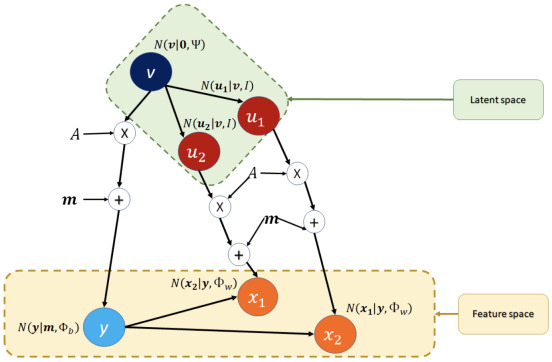


Figure 6: Working schematic for p-LDA[31]

space. Here the variables are considered to be independent[31]. The values for x_1 and x_2 are transformed to u_1 and u_2 in the latent space respectively through some transformation function A to the latent space. The values for x and u and be transformed back and forth through this transformation. This has can be seen in Figure 6. This class association can also be used for identifying objects from different images thus providing a cost-effective and faster method of place classification if used in this manner. However, this modelling technique is centered around Clustering of classes and is an unsupervised technique. It also assumes that all the data is derived from the same distribution but this may not always be the case.

2.3 Spatially coherent Latent Topic Models (LTM)

Spatially coherent Latent Topic Models or spatial-LTMs is probabilistic used for identifying objects within the images through the use of classification as well as segmentation at the same time. It is based on the traditional bag-of-words approach in Topic Modelling and is build upon the previous Topic Modelling methods such as LDA, LSA and pLSA. The advantage of this method over LDA and the other methods is that it segments the different objects in the image and clusters the similar objects through the means of over-segmenting the image. Each object is this method is identified through a homogeneous appearance and patches made by this over segmentation. LDA assumes that each object in an image is an independent patch given a latent topic of the object. This may not always help in effective place classification. This is rectified by spatial LTMs. Under spatial-LTMs,'Only one single latent topic is assigned to the image patches within each region, enforcing the spatial coherency of the model[11]'. This method can help classify and segment objects even when some of them might be occluded or present in multiple occurrences. This technique can be employed as a supervised as well as a unsupervised method. In the case of a supervised method, the partial labels for the different objects in the image can be provided. The underlying technique for spatial LTMs can be found in Figure

7. Here the image is segmented into multiple regions r in the total region space

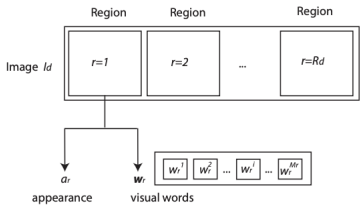


Figure 7: Working schematic for spatial-LTMs[11].

R_d where each homogeneous region in r is specified by an identifier a_r and set of words w_r which serve as objects for this appearance. This is done for each region in the image. For an image, the latent topics z_r are used to associate the different patches in the image to a particular topic. A specific color in a patch represents a visual bag of words for the latent topic to observe and create the topics for the data. The segmentation on an image can be viewed in Figure 8 Additionally, the association of a patch of an image to a particular topic can

Figure 8: Segmentation in the image of a goat to show the homogenous regions in an image to represent an object[11].

be improved by specifying the position of the topic labels and their positioning in an image. The spatial-LTM model performs better than the existing bag of words approach followed by LDA [11].

3 Methods used

This section aims to provide an overview of all the different terminologies which play a part in the project.

3.1 Deep Learning

Deep Learning is a branch of Machine Learning. This field comprises of devising models composed of different layers. These layers when trained on some data help to classify supervised learning tasks such as classification or predict the label of data in the case of unsupervised learning such as clustering or regression. These models are deep in the sense that they are made up of a lot of different types of layers which help store patterns found in data in different layers of the model. These abstractions stored in the different hidden layers are learnt by the model itself. These abstraction layers are connected to each other through

connections. The intermediate layers usually include some Convolution layers which are partially connected, followed by Fully Connected(FC) layers. The top of the layer is called an output layer which is usually a softmax or a sigmoid layer which in turn gives the output as probabilities of the input belonging to a certain class from the labelled data(in the case of classification). A comparison between the workflows of Machine Learning and Deep Learning in Binary Image Classification is illustrated in Figure 9.

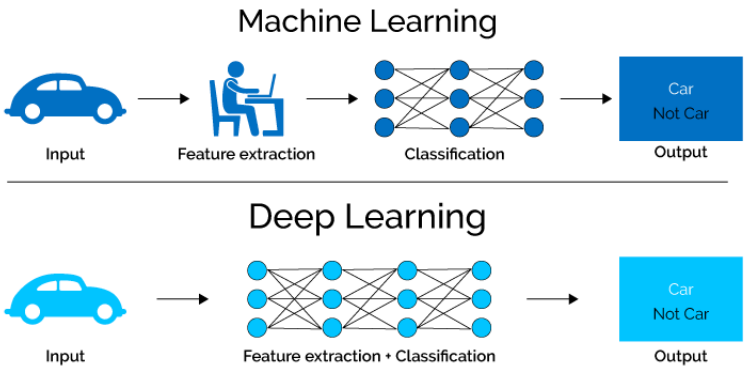


Figure 9: Deep Learning vs Machine Learning[5]

3.1.1 Convolutional Neural Network(CNN)

A Deep Learning Model can be termed as a Convolutional Neural Network with a large amount of layers which is trained to perform a designated task for which it is trained. A schematic for a Convolutional Neural Network(CNN) can be seen below:

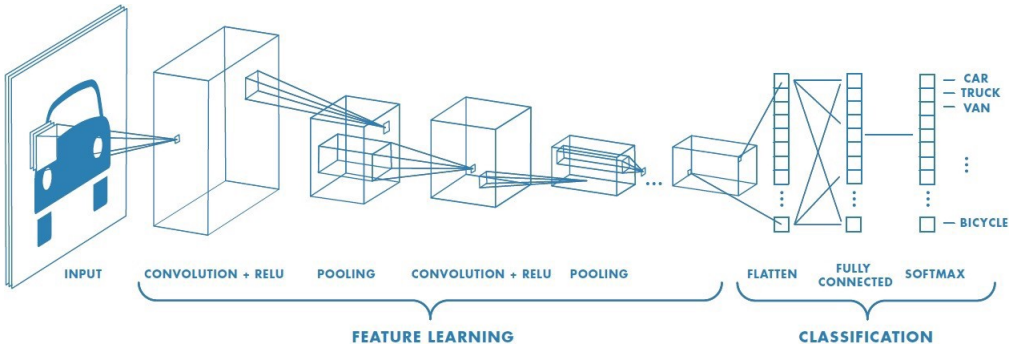


Figure 10: An example of a CNN[29]

From Figure 10, one can visualize the different types of layers that together

make up a CNN. Each layer of the CNN is responsible for a specific task. As CNN offers features such as Convolutional Layers and weight sharing, these have been a major improvement over the traditional Feed-Forward Neural Network architectures of the past being less computationally expensive than the latter[20]. In the example provided in the figure, an input image of a car is fed to the CNN which in turn breaks the image down through the means of Convolution layers followed by a Rectified Linear Unit Layer(ReLU). The output from the ReLU layer is then fed into a Max Pooling layer. This is then followed by a Flatten layer which helps to streamline the outputs from the Pooling layer and then feeds it to a Fully Connected(FC) layer. The output of the FC layer is connected to a Softmax Layer consists of n units where n is the number of classes in the data. In this manner, an input image is converted into class probabilities as output. The different types of layers in the CNN are explained in detail below:

Convolution Layer : CNN is built on the basic foundation of Convolutions which refers combining two different inputs in order to obtain a third input. CNN performs this with the help of a kernel which makes up a feature map. This kernel consists of three dimensions: the height, width of the kernel. Kernels are also called filters. Filters and kernels are usually used interchangeably. 3-D filters are usually used for colored images where the third dimension is the number of channels.

A convolution operation between the filter and the image is performed by moving the filter over the input. Over each element, matrix multiplication is performed over the image and the kernel, and the sum of the output of the convolutions is added to the feature map. The movement of the kernel over the input is guided by the stride value which is essentially the step size for the kernel to slide over the image. A schematic for the above operation is provided in detail in Figure 11.

ReLU Layer : This layer is also called Rectified Linear Unit Layer(ReLU). The output of the convolution layer is fed into this layer. This layer comprises of an activation function which transforms the output from the convolution layer to non-linear which is required by the Neural Network to function.

Pooling Layer : The output from the ReLU layer is fed into a Pooling layer which is responsible for reducing the dimensionality of the inputs and makes it invariable to noise. This also helps prevent over-fitting the model. The most common pooling layer is the MaxPooling layer which takes the maximum value from each feature map window. Thus the feature map is reduced while retaining the important features. This is illustrated in Figure 12

Classification Layers : As shown in Figure 10, the final part of the network consists of a set of Fully Connected layers which are fully connected to the

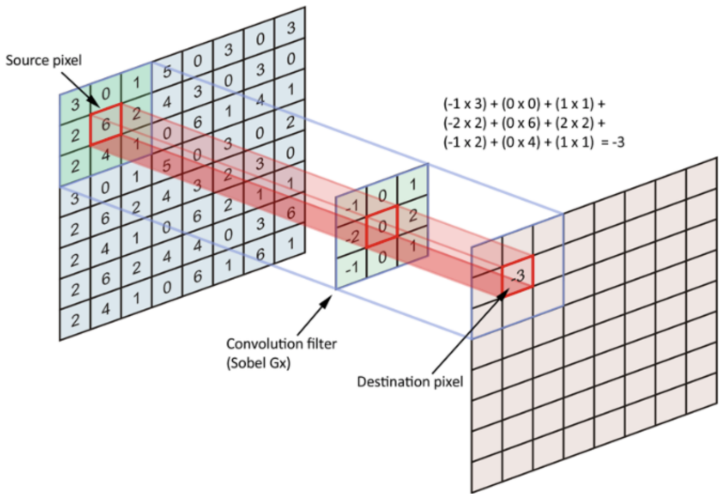


Figure 11: Working schematic of the Convolution Layer[14]

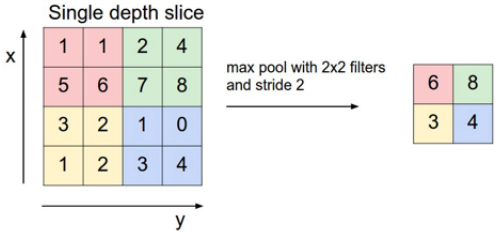


Figure 12: Working schematic of the Pooling Layer(Max Pool)[4]

previous layers. It is responsible for learning non-linear combinations of high-level features resulting from the previous layers. This layer is then connected to another Fully Connected layer which has n units where n is the number of classes required. The activation function for the output layer is 'Softmax' in the case of Multi-Class Image Classification which is required for the scope of the project. Sometimes, a Flatten layer is needed in order to flatten the output from one or more convolution layers before connecting it to the classification layers.

Softmax : Softmax function is also called a normalized exponential function and is used as the activation function for the last layer of a CNN for Multi-Label Image Classification. It transforms the output from the classification layers into class probabilities.The softmax normalizes the class probabilities to $[0, 1]$ which can then be used for prediction. The formula for the softmax function is given

by:

$$\sigma = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{1}$$

where σ lies in the range of $[0, 1]$ and i ranges from 0 to n where n is the number of classes, z refers to the one-hot encoded variables.

Hyper Parameters of a CNN :The Convolution layer in a model depends on the following Hyperparameters:

- Width, Height, Stride of the filter, number of filters used and the padding

Weights and Biases : The crux of the learning of a model is based on setting the values for the weights as well as the biases in the model layers. When a layer receives an input from the previous layers, the weight and the biases values are set with respect to the input as provided by the equation given below:

$$Output = \sum weight * input + bias \tag{2}$$

The above weights are updated with respect to the loss of the model through Back-propagation of Error[24]. The weights and biases of a model are defined as below:

- **Weights** : Weights are responsible for the strength of the signal between two neurons. This is responsible for dictating the influence of a neuron with respect to the input value
- **Bias** : These are additional weights which make sure that there is activation in the neuron even if the kernel weights are set to zero. They are not impacted by the output from the input layers. However, they do influence the output of the neurons.

ReLU layers, Concatenation, Flattening as well as the Pooling layers do not have such weights. Thus these layers do not contain trainable parameters. In the case of Dense Layers(FC Layers), the number of neuron units usually define the number of parameters. In the case of convolution layers, the number of parameters are defined by the following equation:

$$num\_parameters = ((width * height * (filters_{prev} + 1) * filters_{cur}) \tag{3}$$

In the above equation, the 1 corresponds to adding the biases from the previous layers. The *width* and *height* in the above equation refers to the width and the height of the filters respectively.

3.1.2 Examples of Deep Learning Models

The first CNN was developed by Yann LeCun called LeNet in 1989 which consists of layers similar to the one shown in Figure 10. The earlier implementation

of LeNet was used for classifying gray-scale images for zip codes, digits. Over time, several Deep Learning Models have been implemented with slightly altered architectures to suit the task at hand. A few of them are provided below:

- **AlexNet**[3] : A modification over the LeNet which consists of Five convolutional layers followed by Two fully connected layers and 1 soft-max output layer. This network stacks the Convolutional Layers which is helpful in reducing computational bandwidth. It is effective for processing coloured images(with 3 channels for RGB).
- **ZSNet** : This model was implemented in 2013 as a modification to Alex Net. This increases the size of the convolution layers and made the size of the kernel and stride smaller for the previous layers.
- **ResNet**[17] : This model was developed by Microsoft in 2015. This network introduced the concept of skip connections and had extensive application of Batch Normalization. This network is currently considered state of the art for Deep Learning Network. ResNet-50 which is an implementation of this model is used to develop the topic model proposed in the project.
- **GoogLeNet**[33] : This model was developed by Google in 2014 and introduced the concept of Inception Module which helped reduce the number of parameters. This architecture also makes use of Average Pooling. The inception module helps perform a series of convolutions on the input using different kernel sizes and then aggregate the results. Thus making the network wider instead of deeper.
- **VGGNet**[30] :This model was implemented in 2014 and incorporates small convolution kernels of size 3x3 with a stride of 1 as well as adding a max pooling layer of size 2x2 with a stride of 2. Making the network much deeper. This network however, suffers due to the high number of parameters.

3.2 ResNet-50

3.2.1 Introduction

After gaining some basic understanding about Deep Learning Networks and the different types of CNN models, we turn our attention towards the model which forms the primary basis of our image model namely the ResNet-50[17]. As the name suggests, ResNet-50 belongs to a family of Residual Neural Networks.

3.2.2 Residual Networks and Identity Blocks

In order to understand how Resnet works, one must understand the basic concept of a Residual Network which all Resnet architectures are based on. Similar to all the Deep Learning models explained before, Residual networks include

convolution, pooling as well as the classification layers(dense layers). In addition to this, Residual Networks also include identity connection between the layers. These identity blocks help reduce the vanishing gradient problems experienced in previous types of model architectures where-in, adding new layers to the model did little or no change to the accuracy of the model. An example of an identity block can be found in Figure 13.

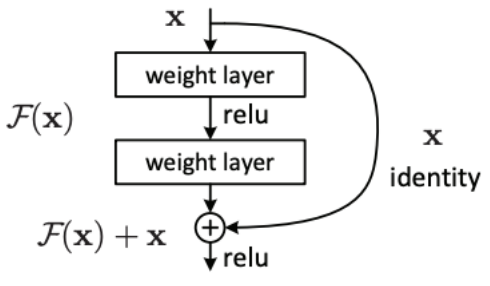


Figure 13: An residual block for ResNet with an identity connection[17].

The residual function can be defined as the difference between the input and the output of the residual block in question. The main function of the residual block is to learn the residual function $F(x)$ such that the accuracy of the network can be increased[17]. Unlike the models without the skip connections and the identity blocks, the model has two pathways in order to update the weights of the previous layers with respect to the loss function. If the weights of the model pass through the pathway with the weights and assign the weights in this block to zero, the second pathway can be used to infer the gradients, thus eliminating the vanishing gradient problem previously encountered. In order for the identity connection to work, the residual block is required to satisfy the following conditions:

1. The identity connection should not contain any parameters. This allows the computational complexity of the model to remain the same.
2. The dimensions for the addition layer for $F(x)$ layer and the identity layer x should be the same.

Thus, through the use of such residual blocks, deep networks with performance comparable to shallow networks can be constructed without adding computational burden on the model.

3.2.3 Resnet-50 architecture

The residual and identity blocks explained above form the crux of the Resnet-50 architecture. The Resnet-50 model consists of 5 stages.

Stage 1 : Stage 1 takes in the input image either as batches of images or a single image(batch size = 1). The dimension of the image is usually set at 224x224x3 where 224 is the height and the width of the image and 3 refers to the number of channels. The first set of layers in this stage are the convolution layers followed by the Batch Normalization layer. The output from the Batch Normalization layer is then fed through a ReLU layer which is then connected to the Max Pooling layer.

Stage 2 : This stage consists of a convolution block which contains 3 convolution layers followed by three sets of Identity blocks which have three convolution layers each.

Stage 3 : The output from Stage 2 is fed to a Convolution block followed by four sets of Identity blocks.

Stage 4 : The image or image batch in this stage is fed to another Convolution block followed by six sets of Identity blocks each with different sizes.

Stage 5 : The output from the previous stage goes through a final set of Convolution block followed by three sets of Identity blocks.

Classification Layers : The output from the previous stages is connected to a 2D Average Pooling Layer followed by a Flatten layer which is then connected to a Dense Layer of 1000 units with Softmax activation. A schematic for the above model can be found in Figure 14. The arrows refer to the identity connections from previous layers.

Batch Normalization : Batch Normalization, also called BatchNorm layers are used to improve the speed of the Deep Learning models by re-centring and re-scaling the inputs. It is used extensively in ResNet-50 for the same reason. The architecture of ResNet-50 can be viewed in detail in Figure 15.
The ResNet 50 architecture defined above will form the base of the Topic Model

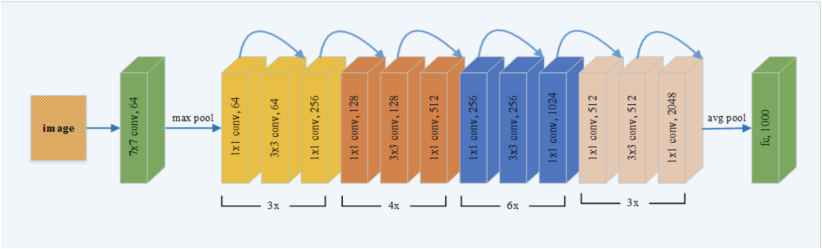


Figure 14: Schematic for Resnet-50 [35]

and the final dense layer of the model will be used in order to derive the list of

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 15: Detailed view of ResNet-18, 34, 50, 101 and 152 along with intermediate output size and shape of the convolution and identity blocks. [17]

topics in order to classify the various places from the dataset. The Resnet-50 model has a total of 50 layers including the input layer as well as the top layer containing the predictions. This model has been implemented in various Python libraries such as Keras and PyTorch as well as in other programming languages such as R, Java and has become state of the art along with VGG-Net for Image Classification.

3.3 Multi-Class Image Classification

Multi-Class Image Classification is one of the most common task which can be performed effectively by a Deep Learning model. Image classification is a supervised learning method which is used to classify the labelled image data into different classes. The model is first fit with the image dataset along with their corresponding labels. The data is first split into three subsets namely:

- **Train** : This is the subset of the dataset which is used to fit the data into the model, which is defined as the training process. Using this data, the model is trained in batches for a number of runs which are termed as epochs. For every batch, the label value is predicted during training and compared to true value of the label. The difference between the predicted and the actual value of the label is termed as a loss term.Categorical Accuracy is also predicted for each batch which is defined as the percentage of correct predicted values for the label. Over subsequent iterations, the weights of the network are updated in order to reduce the loss and increase the categorical accuracy of the model. Once the training process is complete, the training loss as well as the accuracy along with other metrics are

recorded and the training process is set to be complete. In other words, the model has been fit. Once this is done, the evaluation of the model can begin.

- **Validation** : This is the subset of the dataset which is kept separate from the training data and the testing data and is used specifically for evaluating the model. This process refers to testing the model performance with respect to newer data that the model has not come across yet. Weights remain unchanged in this phase. The loss as well as metrics such as accuracy are recorded for the data and observations are made. This helps the user be aware of any instances of model over-fitting or under-fitting. Over-fitting refers to the phenomenon of having a low or minimal training error but having a higher than usual validation error, thus causing the model to make poor generalization on the validation data. Under-fitting occurs when the model has a high training as well as validation loss. This allows the end-user to assess the model and make the desired changes. It is a generally considered a good practice to evaluate the model during every training epoch. However, this may be time consuming in the case of large datasets. Once the user is satisfied by the model results, the model can proceed to the testing phase.

- **Test** : This is the subset of the dataset which is used for testing the performance of the model and making predictions. Testing is usually the final stage of Image Classification. The testing dataset is usually unlabelled i.e. the class labels are not provided to the model during predictions. Once the predictions have been performed, the predicted labels for the dataset are obtained and this can be compared with the true labels of the classes in order to judge the predictions made by the model. Ideal metrics for assessing the predictions are a Confusion Matrix which checks for false positives as well as a Classification Report which consists of metrics such as Precision, Recall as well as f1-score. Unlike the Training and Validation phases, the Testing phase need not require the data to be in batches.

Input Image and Label Shape : The input data fed into the model for Image classification are tensors with rank 4. This is paramount for the model to work properly. The structure of the tensor is given as below:

$$input\_shape = [batch\_size, image\_width, image\_height, num\_channels] \quad (4)$$

$$label\_shape = [batch\_size, categorical\_labels] \quad (5)$$

For example : For a ResNet model with batch size of 1 and image dimensions of (224,224,3) and $number of classes$=5 where the labels belong to classes 1,2,3,4,5: $input\_shape = [1, 224, 224, 3]$ and $label\_shape = [1, [0, 0, 1, 0, 0]]$ where the image belongs to $label$=3 depicted as one-hot labels.

Here the number of channels refer to whether the image is gray-scale(channel

=1) or colored(channel = 3) which could be rgb or hsv. One must make sure to provide the correct ordering of the channels and the input tensor for correct classification of images. In the case of Binary Image Classification, the Output layer is a Dense or a Fully Connected layer with 1 unit which has value 0 or 1 signifying whether the image belongs to class 0 or class 1 as there are only two classes. The output layer has an activation of sigmoid in the case of binary classification. However, in the case of Multi Class Image Classification,the output layer has a softmax activation with n units where n signifies the number of classes. For multi-class image classification, the loss function used can be either Sparse Categorical Cross Entropy or Categorical Cross Entropy. Sparse Categorical Entropy is used when the labels are encoded as integers ranging from class 0 to class $n-1$ for n classes. In the case of Categorical Cross Entropy, the labels are one-hot encoded for n number of classes. Categorical Cross Entropy has been utilized for the scope of this project thus this will be referred to as the loss function used from now on wards in the report.

The mathematical formula for **Categorical Cross Entropy** is given as :

$$CE = -\sum_{i=0}^{n-1} y_i * log(\hat{y}_i) \tag{6}$$

Here $\hat{y}_i$ refers to the value of the label predicted by the model and y_i refers to the target value i.e. the true value of the label for a particular input image. In the case of Multi-Class Image Classification, one image can have only one label. If the input image has more than one label, it is termed as Multi-Label Image Classification. For Multi-Class classification, the sum of all the class probabilities should sum to 1 for this to work.

3.4 Topic Modelling

In simple terms, topic Modelling is a statistical and probabilistic method for detecting topics from a cluster of documents. Topic Modelling is currently applied extensively in text data in order to create arbitrary number of topic where-in for each topic, different words from the documents are assigned to a particular topic based on a probabilistic weight for each word in each topic. Topic Modelling follows the concept that each document is made up of a set of highly occurring topics and each of these topics may include a set of words from either documents. Words belonging to each topic are assigned a probabilistic weight which determine how strongly the word belongs to a particular topic. Along with its extensive application in Text Mining and Sentiment Analysis for text data, it has also found some profound application in detecting topics from image data as well as in the field of bio-infomatics and Computer Vision. Topic Modelling is based on latent distributions such as Latent Dirichlet Allocation(LDA) or Latent Semantic Allocation(LSA) which are explained below.

3.4.1 Introduction to Latent Dirichlet Distribution(LDA)

'Latent Dirichlet Allocation or LDA is a statistical tool which is used in Natural Language Processing'[8]. It is a statistical model which consists of a set of unobservable groups which consist of a set of observations. Each of these observations are bound by a probabilistic value which determines how strongly each of the observation belong to each of these groups. Like the entire premise of Topic Modelling, LDA works under the assumption that every document consists of a set of regularly occurring words and some of these words or observations occur more frequently in some topics more than others. LDA is processed with the help of a Term-Document Matrix. The main hyper parameter for LDA to function is the number of topics specified. In the pre-text of Text Mining, for the LDA model to work, it undergoes four basic assumptions:

1. Each document can be made up of one or more topics which in turn consist of one or more words belonging to the topic
2. Some words in one topic could be ambiguous and having a specific probability value for the topic. However, with the help of the neighboring terms in the document, the word in the specified topic could be made as non-ambiguous.
3. Most of the documents contain only a small number topics. Thus this establishes the basis that some documents may contain more topics than others based on the content of that specific document.
4. Some of the terms in a topic can occur more frequently than other terms in the topic.

3.4.2 Pre-requisites for performing LDA

In order to understand how LDA processes a corpus of documents, one needs to create a TF-IDF matrix which consists of two statistical values namely, Term Frequency(TF) and Inverse Document Frequency(IDF)[26]. These terms are defined below:

Term Frequency (TF) : The TF metric defines the importance of a particular word in a given document among a corpus of documents. It can be explained by the following formula :

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{7}$$

Here, the term frequency is defined as the number of occurrence(frequency) of the term t divided by the frequency of all the terms in a document d[26].

Inverse Document Frequency (IDF) : The IDF checks the amount of information that a word contains in a corpus. It is a measure of whether the term is commonly or rarely occurring across all the documents in the corpus. This term is logarithmically scaled. It is given by the formula specified below:

$$idf(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|} \tag{8}$$

[26] Here N refers to the total number of documents, d refers to the document which contain the term t and D is the set of all the documents to which d belongs[26].

TF-IDF : TF-IDF is defined as the product of the Term Frequency and the Inverse Document Frequency as is given by the following formula :

$$tf - idf(t, d, D) = tf(t, d) * idf(t, D) \tag{9}$$

[26] A high value for the tf-idf correlates to a high TF value and a low IDF value[26]. This means that the term occurs very frequent in one document and this occurrence is quite sparse in the collection of documents. Thus this helps in filtering out the common terms across all the documents. A sample TF-IDF matrix is specified in Figure 16.

The TF-IDF given by Figure 16 is generated from a set of Four Documents

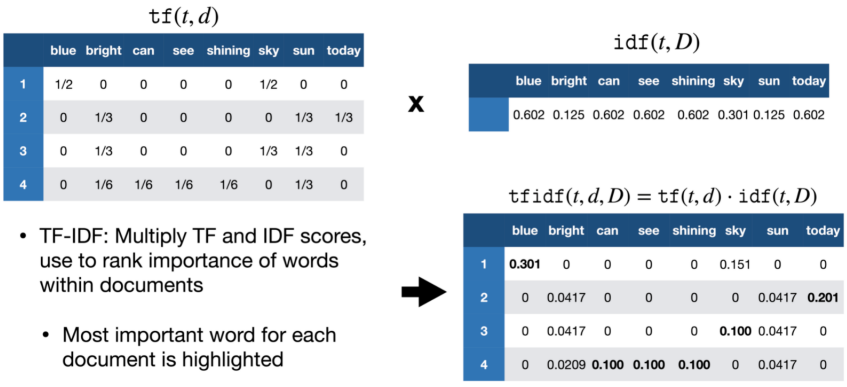


Figure 16: A sample TF-IDF matrix with rows as set of documents and columns as terms)[2]

listed as 1,2,3 and 4 and consists of some common terms which are listed as columns. Each value in the matrix corresponds to the TF-IDF value and is calculated from the formulas listed above. The terms in order to be processed by the TF-IDF matrix are required to undergo pre-processing which can be summarized in three basic steps, namely:

- **Tokenization** : This step involves removal of non-words such as punctuation.
- **Stop Words Removal** : This step transforms all the words into lower case as well as removing the commonly occurring words such as articles(a, an, the) and some common words such as this, is etc.
- **Word Stemming and Lemmatization** : Word Stemming helps to reduce the words into their stem form such as transforming words such as clouding, clouds, cloudy into a basic stem i.e. cloud. Lemmatization maps the different stems of the words into their dictionary form.

There are currently text processing libraries in different programming languages which allow the user to apply the aforementioned pre-processing steps onto the data. This includes the NLTK library in Python.

3.4.3 Application of LDA

Along with the parameter specifying the number of topics, LDA also requires two parameters namely α and β.

- α : α is responsible for allocating the topics to different documents. A high value of α will assign more topics to a document whereas a low value of α will assign fewer topics to a particular document in the corpus.
- β : β is responsible for topic similarity in the model. A high value of β assigns more words to a particular topic whereas a low value of β would assign less number of words per topic.

The functioning of the LDA model[8] can be seen from Figure 17. Along with α and β defined above, there are some other parameters which the LDA depends on and are explained below:

- θ_i : Topic distribution for document i.
- ϕ_k : Word distribution for topic k.
- $Z_{i,j}$: Topic for the document i containing word j.
- $W_{i,j}$: Word j in document i.
- **N** : Number of total words in Document i.
- **M** : Total number of documents.

$$p(w|\alpha,\beta) = \int p(\theta_d|\alpha)(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n,\beta))d\theta \tag{10}$$

[8]

$$p(D|\alpha,\beta) = \prod_{d=1}^{M}\int p(\theta_d,\alpha)(\prod_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta)d\theta_d \tag{11}$$

[8] Equation 10 refers to the marginal distribution of a document. Equation 11 gives the marginal distribution of the entire corpus[8].

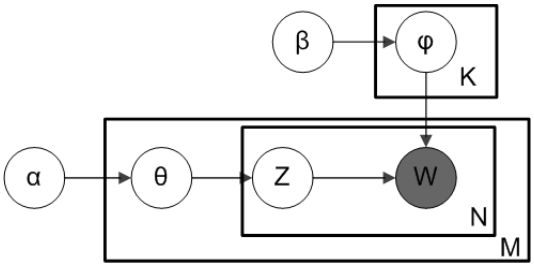


Figure 17: Plate based model for LDA)[9]

3.5 Transfer Learning

In simple terms, Transfer Learning is defined as the task of using a pre-trained Deep Neural Network and re-use the weights of the model in order to perform a secondary task. This helps in improving the performance of the Neural network by reducing the training time of the model when training it from scratch. Transfer Learning is widely adopted in Classification tasks in Computer Vision and Imaging.

In Transfer Learning, the weights of the pre-trained network architectures are preserved when being applied to the problem statement at hand. Network architectures such as VGGNet as well as Resnet trained on massive datasets such as the ImageNet dataset can be re-used and re-purposed by eliminating the final classification layers and replacing them with the modified Network in order to perform the Deep Learning Task at hand. In this project, a pre-trained Resnet-50 architecture is used to train both the Baseline as well as the Topic Model architectures by fixing the pre-trained weights and only training the top layers of the network. Here the top-layers refers to the classification layers for classifying the dataset images to 24 classes for EgoPlaces dataset as well as 365 classes for Places365 dataset. The pre-trained Resnet-50 model is trained on 1000 classes from the ImageNet dataset.

The steps involved in Transfer Learning are specified below:

1. **Model Selection** : Selecting a model which has been pre-trained on a large dataset for a problem similar in nature to the task which is required to be performed by the user. These pre-trained networks with weights are included in various libraries and are available for use by various institutions.

2. **Reusing the model** : Once the Pre-trained model is short-listed by the user, the model can then be used as a starting point for performing the Classification/Regression task required.

3. **Fine-Tuning** : The final step of Transfer Learning involves fine-tuning the final network in order to obtain a model for the dataset of interest. The weights of the entire model here are set to be trainable and the final model is obtained which has the minimum loss as well as the highest accuracy

which will be tuned for the final task to be performed. Here the final task refers to performing Image Classification on the EgoPlaces as well as the Places365 dataset.

Figure 18 illustrates the entire workflow for Image Classification. Here the model is first pre-trained with the source data with multiple classes. This model is then re-used and re-purposed with the target data for performing binary image classification.

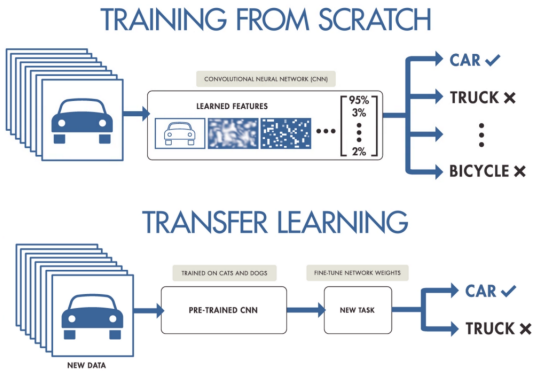


Figure 18: Workflow for Transfer Learning for Image Classification [32]

3.6 Python Libraries

3.6.1 Introduction

The programming language used for the project is entirely in Python. This section deals with the libraries in Python which are required to code the project ranging from creating the dataset to plotting the metrics as well as defining the different model architectures.

3.6.2 Numpy

Numpy, also called NumPy is a Python library which helps in performing high level operations through the support of multi-dimensional arrays[16]. It was first developed in 1995 and came to be used in Python since 2006. This library is extensively used in fields of data science. The main feature of Numpy is the n-dimensional array structure which are used to encode various operations such as Fourier Transforms as well as Linear Algebra and other mathematical operations in Python. Numpy operations are GPU and multi-processor enabled and is easy to use and scale operations. NumPy helps enable the use of other Python scientific libraries such as OpenCV, Pandas, Scikit as well as many others. With the help of NumPy and its extensions tasks such as Signal Processing, Image Processing as well as Mathematical Analysis can be performed on multi-dimensional data.

3.6.3 Pandas

Pandas[22] is an open source data processing tool-kit which is built on top of NumPy. It includes complex homogeneous data structures such as Series as well as heterogeneous data structures such as Data Frames which find vast application in data analysis. It is available in major programming languages such as Python as well as R. The main task of this library is performing data manipulation on Data structures as well as Time Series data. Operations on such data structures such as Data Frames and Series data include major database operations found in SQL based languages. Data obtained from CSV as well as JSON based files can be processed using Python. It includes easy to use functions which help perform data wrangling as well as reshaping, merging, selecting and cleaning the data present in Data Frames and Series.

3.6.4 SciKit-Learn

SciKit-Learn[25] is a Python based library based upon SciPy, Pandas and NumPy. SciPy is the broader library which helps Python perform scientific computing on different types of data. Scikit-Learn is a part of the SciPy eco-system. Scikit-Learn includes mathematical tools which are used for performing Machine Learning in Python and includes functions which are used to perform Data Science tasks such as Classification, Clustering and Regression. It also helps perform tasks such as Model Selection, Data Pre-Processing as well as Dimensionality Reduction. Model Selection includes Cross-Validation as well as defining the metrics for validating the data models such as 'accuracy', 'precision', 'recall', 'f1-score' and many more. Pre-processing includes functions responsible for Standardizing and Normalizing the data. The types of models included in this Python package are basic models such as K-Means, K Nearest Neighbors as well as provide support for Support Vector Machines (SVM). It also provides metrics for predicting the model output such as Confusion Matrix as well as Classification Report.

3.6.5 Matplotlib

MatPlotLib[18] is a plotting library based in Python and has similar functionality as MATLAB. This library includes API based functionality required in order to tweak graphs including adding axes values and plot titles. It makes use of NumPy arrays in order to process and the plot the data. The different types of plots offered by this Python library are given below:

- Line plot, 3D plot, Scatter plot, Image plot, Contour plot, Histogram, Bar plots[18] and many more ...

3.6.6 Seaborn

Seaborn[36] is another plotting library in Python. It is derived from Matplotlib and provides tools for creating statistical data visualizations. It enables

complex visualizations unlike Matplotlib and provides full support for Pandas DataFrames. The plots generated by the Seaborn library have a greater visual appeal as compared to MatPlotLib. Some of the visualizations provided by this library are listed below:

- Relationship Plots(relplots), Composite views for Multi-Variate datasets such as joint-plots, Treemaps as well as Heatmaps[36] and many more ...

3.6.7 TensorFlow

Tensorflow[6] is a open source software library released by Google. It is widely used for the training and evaluation of Deep Learning Models. It is based on data flow and differential programming. The recent version of Tensorflow is 2.0 and is being widely adapted for low level as well as high level APIs pertaining to Machine Learning due to its ease of use as compared to the previous versions. It is a good substitute to PyTorch which was the state of the art before Tensorflow 2.0. It is programmed in C++ and is available in various programming languages such as Java, C++, C#, R, Python and many more. It helps in performing computations at scale on GPUs through the use of PyCUDA library in Python and has good support for Multi-Processing. TensorFlow makes use of complex multi-rank multi-dimensional arrays called Tensors which allow the end-user to customize low as well as high level API tasks with much ease. It has found applications in various fields of data science such as Classification and Regression on Time-Series, Image, Video as well as Audio data.

It allows the end-user to view run-time statistics such as Model fitting, viewing model metrics as well as assessing the loss of the model through TensorBoard.

TensorFlow 2.0 offers full support for Keras in order to perform high level operations on the Deep Learning Network. It also offers full support for dataset generation and model fitting through the use of tf.dataset which helps drastically improve the performance of Deep Learning models. Features of tf.dataset include interleaving the dataset, creating data shards, caching through pipelines which can also be visualized using TensorBoard.

3.6.8 Keras

Keras API[12] is the open-source library used for creating Deep Learning models. It runs on top of TensorFlow. Keras is entirely responsible for creating such models through the use of directives such as models and layers. The model functionality is defined for processing simple to complex models which can be defined sequentially or through the use of an functional API which thus helps build the model in a graph-like manner. A model is built on top of layers. A simple model can be defined in the following manner:

```
import tensorflow as tf
inputs = tf.keras.Input(shape=(3,))
```

```
x = tf.keras.layers.Dense(4, activation='relu')(inputs)
outputs = tf.keras.layers.Dense(5, activation='softmax')(x)
model = tf.keras.Model(inputs=inputs, outputs=outputs)
```

[12]

In the above example, a model is defined as having the input shape of 3 containing a hidden layer of 4 units with ReLU activation. The output of the hidden layer is connected to a dense layer with softmax activation containing 5 neurons. The final line of code creates the actual model. The model statement takes in the input and the output layers as parameters. Thus a simple Neural Network model is defined using just 4 lines of code.

The model layers can be considered as tensors which in turn include the following parameters:

- Weights, biases, loss function, units or neurons contained in each layer, regularizers such as L1, L2, L1-L2 regularizers, Activation functions such as Softmax, ReLU, Sigmoid, Weight initializers such as random, random-normal and many other functions

It is not important to specify each of these parameters individually as some of them are automatically inferred. However, some of the parameters are important such as the number of units of each layer as well as the input shape in the case of the input layer. The final layer is a softmax activated dense layer which should contain the number of classes as the number of units to be used.

Keras also offers pre-trained model architectures such as VGGNet, Inception as well as ResNet architectures from the box and helps the end-user to build the desired models with ease. In the case of Image Processing, it includes libraries such as ImageDataGenerator which in turn helps to process the data from images into NumPy arrays from file systems as well as csv files to be fit by the deep learning model using limited lines of code. Models either Sequential or Functional can contain custom layers or ready made layers such as the following:

- **Convolution Layers** : Convolution3D, Convolution2D, Convolution1D etc.
- **Pooling Layers** : GlobalAveragePooling, MaxPooling3D, MaxPooling2D etc.
- **Dense Layers** : Fully connected layers
- **Regularizer Layers**
- **Preprocessing Layers**
- **Normalization Layers**
- **Recurrent Layers** : LSTM, GRU layer etc.

The main advantage of Keras is that it helps the user to define complex models with much ease as well as define the various parameters for each layer such as regularizers as well as how the weights should be initialized. Once the model is created, the user can view the model summary through *model.summary* as well as plotting the model through *plot_modelmodel_name*. Along with using pre-defined models and layers, it also allows the user to define custom model and layers through the means of sub-classing. Definition of custom layers and models, however is limited to only the functional API mode and not sequential. A custom layer is required for the scope of the project and will be explained in the following section.

3.7 K-Fold Cross Validation

K-Fold Cross Validation is a method to evaluate the performance of a model with limited data[27]. This method involves the parameter k which refers to the number of groups in which the data is split thus attributing to it being called k-fold. If $k = 10$ then this procedure is called a 10-Fold Cross Validation procedure. This method is widely adopted to assess the performance of the model on unseen data. The advantages of K-fold CV is that it is easier to understand and helps in fitting models with less bias than one would get with a simple test-train split of the data. The methodology of K-Fold Cross Validation is as follows:

1. The entire dataset is shuffled in order to get a proper distribution of the dataset to each fold or group.
2. The shuffled dataset is then split into k groups depending on the value of k.
3. For each fold :
    - One fold of the data is chosen as the hold-out set which will be used to evaluate the model.
    - The remaining folds are chosen as the Training set for the model.
    - The model is then fit with the training set and then evaluated for the test set.
    - The evaluation metric scores for the models are then retained and the model is discarded.
    - The same procedure as above is repeated for the subsequent folds.
4. The evaluation scores for each fold are then displayed and the model with the highest performance for a particular metric is chosen.

Each fold of the dataset should preferably have the same size in order to obtain proper results. A special case of K-Fold Cross Validation is when the value for k is set to n where n is the number of samples in the complete dataset. This method is called Leave one out Cross-Validation(LOOCV). Through the

scikit-learn library, the implementation of K-Fold is readily available in Python.

A detailed schematic for the K-Fold Cross Validation procedure can be seen in Figure 19.

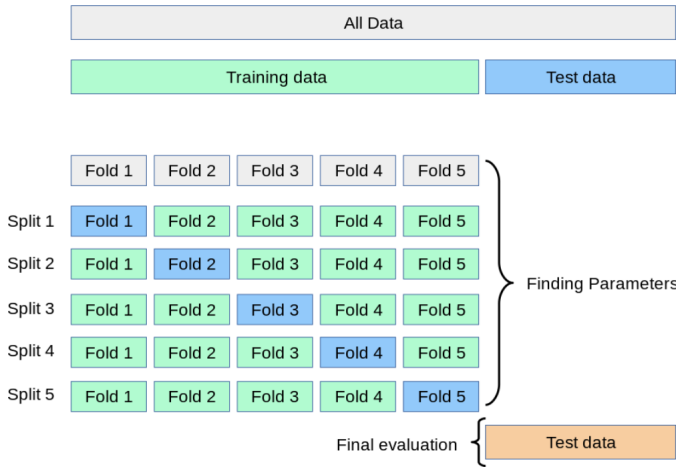


Figure 19: Workflow for K-Fold Cross Validation with 5 folds($k = 5$) [1]

3.8 Stratified K-Fold Cross Validation

Stratified K-Fold Cross Validation[7] is a variation of the K-Fold Cross Validation method for model selection and evaluation. The difference in stratified k-fold is that each fold is a representative of all the strata of data. In the case of multi-class models, this means that each fold contains samples having all the class labels. Thus this helps in training models with folds having lesser mean and variance than a regular K-Fold Cross Validation. The downside to this procedure is that the value for k in this method is equal to the maximum permissible number of splits allowed on the label class having the lowest samples in the dataset.

For Example : If there is a dataset with 5 classes with one class only containing 3 samples in the entire dataset, the value of k can not exceed 3, allowing only 3 folds from the entire dataset. The procedure for Stratified K-Fold is similar to regular K-Fold albeit the manner in which the data is split. A schematic for the working of Stratified K-Fold Cross Validation can be seen from Figure 20.

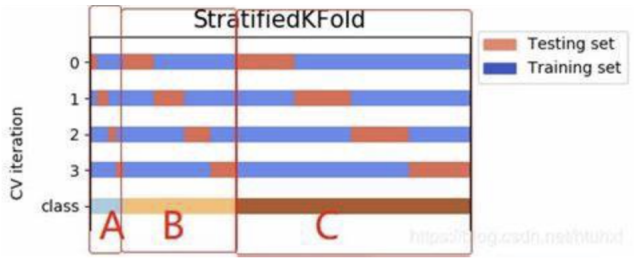


Figure 20: Workflow for Stratified K-Fold Cross Validation with 4 folds($k = 4$) and three classes A,B and C [23]

4 Datasets

4.1 Introduction

This section deals with the different databases used for testing the baseline as well as the proposed topic model. There are two databases which are used for fitting the model namely EgoPlaces and Places365.

4.2 EgoPlaces

4.2.1 Introduction

EgoPlaces is a part of the greater EgoRoutine dataset. The EgoRoutine dataset was first used as a dataset for behaviour categorization used in EgoCentric photostreams[34]. This dataset was created by the Computer Vision and Machine Learning at the University of Barcelona(CVMLUB) consolidated research group(2017SGR1742)[34]. The EgoRoutine dataset contains the day to day activities of seven users. The EgoPlaces dataset is responsible for categorizing the images for the users into places which are specified by a list of excel files, one for each user.

4.2.2 File Structure

The EgoPlaces is made up of about 120,000 images divided among the seven users ranging from *user_01* to *user_07* folders. Each of these user folders contain sub-folders which enclose the activities of each user. The distribution images representing the activities of these seven users can be summarized in Table 1.

4.2.3 Class Labels (Places)

The different place categories for the EgoPlaces dataset are summarized in Figure 21.

The labels for the user images are provided by excel files. There are in total 7 excel files for the 7 users where each sheet of the excel file contains the labels for the activities of the users. The excel files are listed as:

User #	Number of days	Total Images per user	Average images per day	Std dev
User 01	14	20527	1467	183.9
User 02	10	9594	960	226.87
User 03	16	21608	1351	405.65
User 04	21	18977	949	125.31
User 05	13	17046	1311	262.46
User 06	18	15535	971	311.54
User 07	13	11185	860	182.3
Total		**114,472**		

Table 1: EgoPlaces dataset distribution for the number of users and their respective activities into days

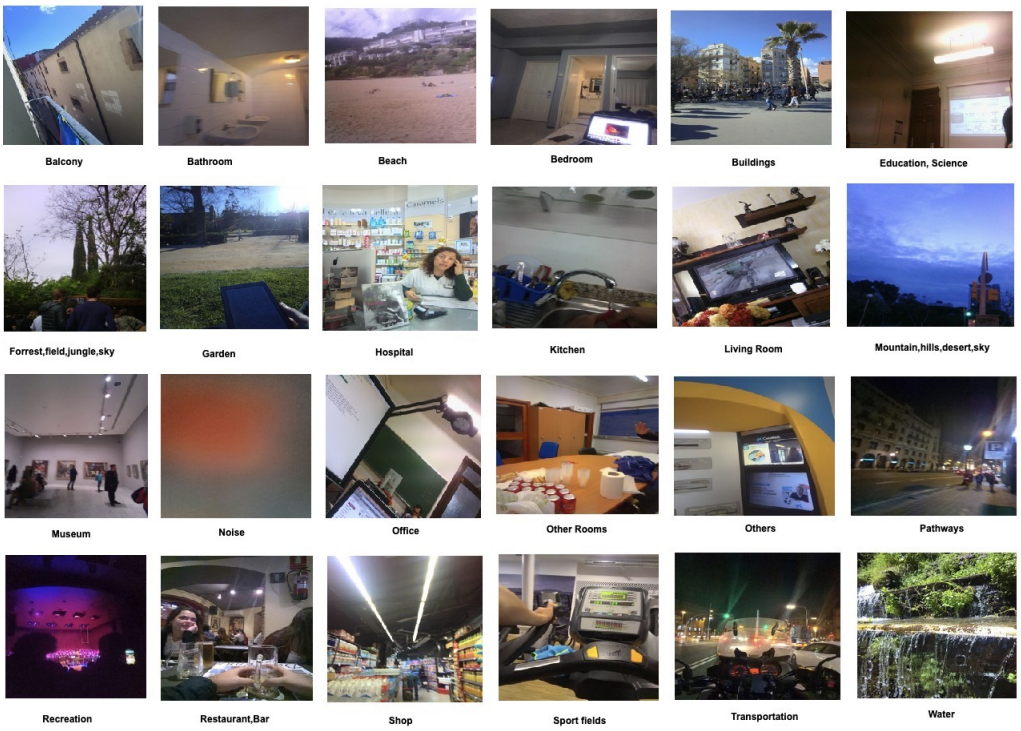


Figure 21: Place categories for EgoPlaces dataset[34]

- user_01.xlsx, user_02.xlsx, user_03.xlsx, user_04.xlsx, user_05.xlsx, user_06.xlsx, user_07.xlsx

The naming convention of the sheets for the excel files are synonymous with the sub-folders of the dataset. An example of the excel sheet containing the labels can be viewed in the following table :

	Photo	Class
0	20171201_115957_000.jpg	**Bedroom**
1	20171201_120015_000.jpg	**Bedroom**
2	20171201_120032_000.jpg	**Bedroom**
3	20171201_120049_000.jpg	**Bedroom**
4	20171201_120106_000.jpg	**Bedroom**

Table 2: Table for the excel sheets containing the Image name and its corresponding Label

There are a total of 24 distinct labels for the entire image datasets for EgoPlaces. These are summarized in Table 3.

Place Categories		
Balcony	Hospital	Others
Bathroom	Kitchen	Pathways
Beach	Living room	Recreation
Bedroom	Mountains, hills, desert, sky	Restaurant/Bar
Buildings	Museum	Shop
Education/science	Noise	Sport fields
Forest, field, jungle	Office	Transportation
Garden	Other rooms	Water

Table 3: 24 distinct places for the EgoRoutine dataset

4.2.4 Data Pre-Processing

In order to utilize the EgoPlaces dataset for the different models architectures defined in the project, it was required to reshape the dataset to one which can effectively perform Image Classification on the proposed baseline model as well as Topic Modelling for the proposed topic model.

The steps that are required for Pre-Processing the EgoPlaces datasets are provided in detail below:

1. Take all the images from the different users into a single common dataset. This common dataset includes the entire list of images from the activities for all the users.

2. Obtain the labels for each image from the user excel data files into a single file. This single file contains the labels from the different sheets corresponding to each user.

3. Merge the image labels with the single unified dataset containing the paths of the different images in the format dictated by Table 2.

4. Transforming the Image names to include the full paths of the images in a csv file. This includes appending the complete path of the images to the present Image files. This step takes in all the images names from the previously merged file and transforming the Photo column in the format given below:

   ```
   /user_name/activity/Photo name
   ```

   For example an image 20171201_115957_000.jpg belonging to user 01 and activity 1(day 1), the format for the path will be given by:

   ```
   /user_01/user1_01/20171201_115957_000.jpg
   ```

5. Subset the unified dataset by their corresponding labels. This involves grouping the different images by their class which in our case is the places as listed in Table 3.

6. For each of the different categories of places, split the data into three folders namely:

   - **train** containing 60% of images for each place.
   - **val** containing 20% images for each place.
   - **test** containing 20% images for each place.

7. Update the paths for the images to reflect the test, train and val folders followed by their respective places in a new dataset csv file. This will be the data set which will be used in order to process the images. The format for this transformation can be viewed below:

   ```
   /data/{test|train|val}/{place}/Image Name
   ```

   Here the place is a placeholder for any of the classes belonging to Table 3. Taking the previous example image 20171201_115957_000.jpg with Bedroom as the place label, if this image is placed in the Train folder, the new directory-folder structure for this will be:

   ```
   /data/train/Bedroom/20171201_115957_000.jpg
   ```

8. Copy all the images from the previous unified dataset to the corresponding train, test and val folders. Each of these folders contain sub-folders corresponding place names using the csv file defined in the previous step.

9. Resize all the images to size (224,224,3) which is a pre-requisite for using the Resnet-50 base model.

4.2.5 Final Dataset

The final dataset structured from the EgoRoutine dataset can be broken down and summarized in Table 4

Places	Train	Val	Test	New data
Office	20185	2704	6309	9007
Transportation	7880	893	2082	2260
Pathways	6718	862	2009	1862
Restaurant,Bar	6443	847	1976	869
Education,science	4143	552	1287	451
Noise	2478	333	774	238
Living room	2034	282	657	141
Kitchen	1017	135	313	135
Bedroom	911	121	282	121
Shop	844	114	266	108
Other rooms	740	98	228	90
Recreation	460	61	141	87
Forest, field, jungle	441	59	136	60
Mountains, hills, desert, sky	436	59	135	58
Beach	432	58	133	56
Sport fields	300	41	94	55
Bathroom	291	39	91	39
Garden	289	39	88	30
Others	163	22	50	11
Buildings	108	15	33	7
Balcony	97	14	31	4
Museum	53	8	17	3
Hospital	29	5	9	2
Water	3	1	1	1
Total	56495	7362	17142	15695

Table 4: The final breakdown of the Train,Validation,Test and Prediction data folders with the corresponding place labels

4.3 Places365

4.3.1 Introduction

The Places365 database is created based on the principles dictated by human cognition. It is designed in order to train artificial systems for high level visual understanding tasks such as scene classification as well as object detection[39]. The dataset is categorized by places such as airfield to bedroom. The Places365 dataset is composed of 1.825 million images which are categorized by 365 place labels with around 5000 images belonging to each label(place). These different

labels will be discussed in the following section. Additionally, it also includes 36,500 additional images in order to validate the data. The size of the images are (256,256,3).

Places365 dataset was created for the Places365 challenge in 2017. It was supported by National Science Foundation directorate(#1016862), The McGovern Institute Neurotechnology Program(MINT), ONR MURI N000141010933,MIT Big Data Initiative at CSAIL and Google, Amazon, Xerox and NVIDIA[39].

4.3.2 Directory Structure

The directory structure for the Places365 is similar to the final structure of the EgoPlaces dataset. Here the data is split into train, test and val folders. For each of these folders, the image dataset is broken down into 365 folders, one for each class. Some of the images from this dataset are illustrated in Figure 22.

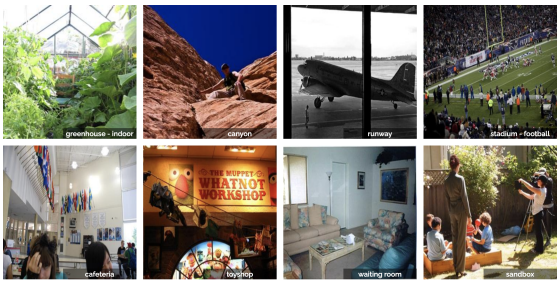


Figure 22: Some of the images from the Places365 dataset with their respective classes [39]

4.3.3 Class Labels(Places)

There are a total of 365 categories of places some of which are summarized in Table 5. The place labels for the rest of the classes can be seen in Tables 47, 48, 49 in Appendix 11.6. For each class, the Places365 dataset contains 5000 images each for train as well as 100 images per class for testing and validation.

4.3.4 Data Pre-Processing

Due to resource constraints involved in processing 1.85 million images, the dataset has been sampled in this project. There are 1500 images per place category instead of the original count of 5000 per image category. The data used for testing the performance of the models on this dataset is also sampled from the training data from the original Places 365 dataset.

The images are scaled to (224,224,3) in order for them to be processed by the backbone Resnet-50 model.

PLACES		
0. airfield	1. airplane_cabin	2. airport_terminal
3. alcove	4. alley	5. amphitheater
6.amusement_arcade	7. amusement_park	8. apartment_building/outdoor
9. aquarium	10. aqueduct	11. arcade
12 . arch	13. archaelogical_excavation	14 . archive
15. arena/hockey	16. arena/performance	17. arena/rodeo
18. army_base	19 . art_gallery	20. art_school
21. art_studio	22. artists_loft	23. assembly_line
24. athletic_field/outdoor	25. atrium/public	26. attic
27. auditorium	28. auto_factory	29. auto_showroom
30. badlands	31. bakery/shop	32. balcony/exterior
33. balcony/interior	34. ball_pit	35. ballroom
36. bamboo_forest	37. bank_vault	38. banquet_hall
39. bar	40. barn	41. barndoor
42. baseball_field	43. basement	44. basketball_court/indoor
45. bathroom	46. bazaar/indoor	47. bazaar/outdoor
48. beach	49. beach_house	50. beauty_salon
51. bedchamber	52. bedroom	53. beer_garden
54. beer_hall	55. berth	56. biology_laboratory
57. boardwalk	58. boat_deck	59. boathouse
60. bookstore	61. booth/indoor	62. botanical_garden
63. bow_window/indoor	64. bowling_alley	65. boxing_ring
66. bridge	67. building_facade	68. bullring
69. burial_chamber	70. bus_interior	71. bus_station/indoor
72. butchers_shop	73. butte	74. cabin/outdoor
75. cafeteria	76. campsite	77. campus
78. canal/natural	79. canal/urban	80. candy_store
81. canyon	82. car_interior	83. carrousel
84. castle	85. catacomb	86. cemetery

Table 5: Place Labels in Places365 Dataset (Part 1)

5 Proposed Methodology

5.1 Introduction

This section entails the Proposed Methodology for this project. The first part of the implementation deals with creating a baseline model for performing Multi-Class Image Classification on the EgoPlaces as well as the Places365 dataset. This will be followed by implementing the different versions of Topic Models for k number of topics. The performance of each of these Topic Model implementations will be compared against the baseline models for the different set of parameters. The model architecture as well as the pipeline for both the baseline and the topic models are given in the subsequent section of the report.

5.2 Data Fitting Strategies

5.2.1 Image Data Generator

The first methodology to be followed in order to fit the input data would be by feeding it through an Image Data Generator. Image data generator is a method which allows the end-user to transform the images required to be fit in the model. The input image data initially is a set of file names which can be obtained either from the particular directory of images(test, train or validation directory) or from a csv file containing the particular file names. Image Data Generator allows the user to augment the input image data by adding random flips or skew to the data in order to prevent it from over-fitting the data and allowing the model to learn more from the same data. One basic augmentation of the data would be to divide the image pixel values by 255 in order to get the pixel vales in the range of 0 to 1. The ImageDataGenerator is invoked by using the following command in Keras and Tensorflow:

```
datagen = ImageDataGenerator(rescale=1./255)
```

Here datagen is an instance of the ImageDataGenerator which helps to rescale the pixels of the image to a scale of $0, 1$. Additionally the different augmentations that are permissible using an ImageDataGenerator would be:

- Random Rotation, Shift, Flips, Brightness, Zoom the different images at random and many more.

The desired augmentations and the re-scaling of the input images from the Image data generators are then applied to the train, test and val sets through the *flow()* method. There are different variations of the flow() method some of which are provided below:

- *flow* : The basic method for creating the data. This takes in the values x and y where x is the input images with rank 4 and y are the corresponding images for the labels. This method takes in the NumPy arrays and generates the batches for the augmented data.

- *flow_from_dataframe* : This method takes in the dataframe containing two columns. The first column contains the file names of the images and the second column takes in the corresponding labels. This method also takes in the path to the directory as a parameter where the images are present in the disk. The flow_from_dataframe method then provides the desired augmented data as batches. The batch size can be specified in the function itself. An example for this method is provided below:

```
trainGen = dataGen.flow_from_dataframe(
dataframe, directory='/data/train',
x_col='filename', y_col='place',
weight_col=None, target_size=(224, 224),
color_mode='rgb',classes=None,
```

```
class_mode='categorical', batch_size=64, shuffle=True,
seed=None,save_format='png')
```

In the above example, the ImageDataGenerator instance, *dataGen* is created for the training data with target size of (224,224,3). The value 3 states that the images are in the rgb form which is specified with the help of the *color_mode* argument. The *class_mode* argument helps to convert the labels into one-hot vectors required by the model if the *class_mode* is set to categorical. In the case of predicting the data, the *class_mode* can be set to $None$. The $shuffle$ attribute helps to randomly shuffle the images and the experiments can be replicated by setting the *seed* attribute to a number. *x_col* is a column of the dataframe which contains the file names of the input images and *y_col* is a column of the dataframe which contains the Label of the image. The names of *x_col* and *y_col* should correctly correspond with the column name of the dataframe being flown to the ImageDataGenerator. The value trainGen can then be applied to the model.fit function directly.

- *flow_from_directory* : This is the method of flowing the ImageDataGenerator directly through the image directory. It has identical attributes as the $flow\_from\_dataframe()$ functionality. However, it does not require the user to specify the dataframe as well as the *x_col* and *y_col* values. An example of this method is provided below :

```
testGen = dataGen.flow_from_directory(
directory, target_size=(224, 224),
color_mode='rgb', classes=None,
class_mode='categorical', batch_size=64,
shuffle=True, seed=None,save_format='png')
```

As one can follow along the aforementioned example, the directory can be specified in the directory argument while the other arguments are similar to one seen in $flow\_from\_dataframe()$. The above example would be to generate the images used to predict the model using model.predict() as the *class_mode* is set to *None*.

5.2.2 TensorFlow Datasets

Tensorflow Datasets are an important feature of TensorFlow which allows the end-user to create APIs in a streamlined fashion. It consists of an API which helps in creating efficient pipelines for fitting the input data into the model in a way which consumes lesser resources than that found in the ImageDataGenerator functionality offered by Keras. It usually involves the following steps to be followed :

1. Creating a source dataset from the input.
2. Apply the dataset transformations to pre-process the data.
3. Iterate over the dataset and then process batches of the pre-processed input.

The tf.data.Datasets are made up of placeholder tensors which process the input data to fit the models on the fly. Thus the entire dataset is not present all the time in memory. The tf.Datasets can be consumed from tensors or read directly from memory or dataframes through csv files. In order to fit the data, the following steps are performed :

1. Reading the input images and their corresponding labels from a csv through a dataframe for train, test and val data.
2. Converting the filenames from the csv files to tensors. This is where the input pandas are transformed to tensors which will be used to fit the model for the different sub-sets of the data.
3. Un-batching the data subset tensors to give tensors containing individual file names
4. The dataset tensors containing the filenames are then mapped to a transformation function. This function is responsible for the following operations :
    - Get the label of the file from the image filename. This usually is the sub-directory of the data subset. For example for the *train* directory, the images for place *Office* are stored in the Office sub-directory. In this way, the place names(labels) for the images can be extracted.
    - Convert the label names to one-hot labels. The order of the labels depend upon the list of places in alphabetical order.
    - Reading the image from the tensor as a NumPy array and specifying the number of color channels. If the number of channels is 3, it is a colored image or it is gray scale if the number of channels is 1.
    - Resizing the images to the desired format of $224, 224, 3$ as desired by Resnet.
    - Normalizing the pixel values of the image in order to get more efficient output from the model. This step would also allow for any other augmentations that have to be performed on the image.
    - The image array and the corresponding label are then rendered as output from the transformation function.
5. Caching the data and storing it to memory. This involves processing the subsets of the data and keeping it in the memory for faster performance. It is advisable to store the data in the CPU if the size of the entire data does not fit to the GPU.

6. Batch the cached data depending on the batch size specified. This dataset is then set to a repeat function in order to repeat the data depending on the number of epochs the model is to be trained for. If the dataset is not repeated, the dataset might get exhausted causing the model to stop fitting. The repeat process can also be performed indefinitely. This is not performed for the test set as the operation is only performed once per model.

7. Pre-fetch the next batch of data. This process can also be parallelized in order to boost the efficiency of the model performance through interleaving and data sharding. The pre-fetched dataset is then assigned to train the model by using the $model.fit()$ functionality.

A schematic for the above-listed operations can be found in Figure 23.

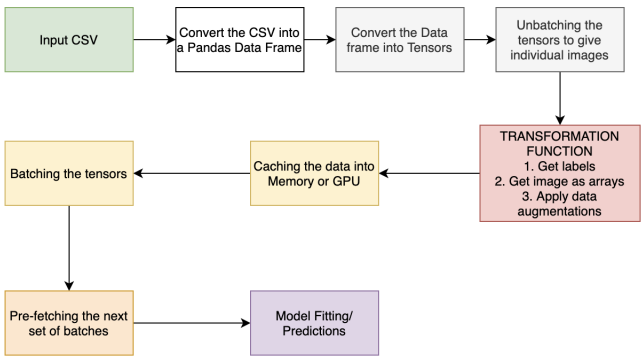


Figure 23: TF Dataset Data Fitting Pipeline

5.3 Baseline Model

In order to test the feasibility and the accuracy of the proposed topic model architectures, they must have some baseline architecture to compare their performance. This is where the Baseline model comes into consideration.

5.3.1 Model Architecture

This model contains the ResNet-50 model serving as the backbone for the model. The ResNet model in this case does not contain the top layer of 1000 units but is replaced by a Softmax layer containing n neurons where n depends on the number of classes present in the dataset. To place things into context, the top layer in the case of EgoPlaces dataset contains 24 units while the one with the Places365 has 365 units.

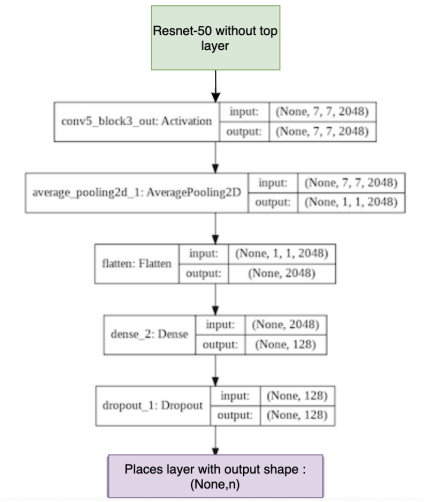


Figure 24: Baseline Model Architecture

The implementation for the Resnet-50 in this model is similar to the one defined in the previous sections (Refer 3.2.3). Without the final classification layers, the output from the Resnet network has the shape $(batch\_size, 7, 7, 2048)$. This layer is then supplied to a GlobalAveragePooling3D layer with a pool size of (7,7). This helps in order to flatten the output for use by the top layers. The output of the Global AveragePooling layer is connected to the final layer with softmax activation containing n units where n is the number of classes. This model is then trained for both the datasets. This is followed by evaluating the model with the testing data. The evaluated model is then used for predicting the output for new data. The model architecture of the Baseline model is given in Figure 24.

For every step of the model training process, metrics such as Categorical Accuracy, Precision and Recall are calculated. This will form the benchmark for training the different implementations of the topic models proposed in the following sections.

5.4 Topic Model Implementations

5.4.1 Implementation 1

The first implementation of the Topic Model could be termed as a naive model implementation. Similar to the Baseline model architecture, the topic model also contains the Resnet-50 model as its backbone. However, in the Topic Model, the top layer from the Resnet-50 layer is included. This top layer is composed of 1000 units. These units serve as the different topics which are evaluated by the model.

The output of the Resnet-50 softmax is fed to four distinct dense layers,

each having a 1000 units each. These four layers are responsible for topics 1 to 4 respectively. The output from these layers are concatenated to a single layer. The concatenated layer is then fed to a flatten layer which is responsible for flattening the output further. This is required in order to classify the images from the datasets with respect to places. This layer is then fed through a series of dense layers. The first layer has an activation of ReLU which is connected to the final dense layer with n units for the different places from the dataset. The dense layers contain dropout layers in between which helps to reduce overfitting in the dataset. The pipeline for this implementation of the Topic Model can be summarized in Figure 25 and the model is defined in Figure 26.

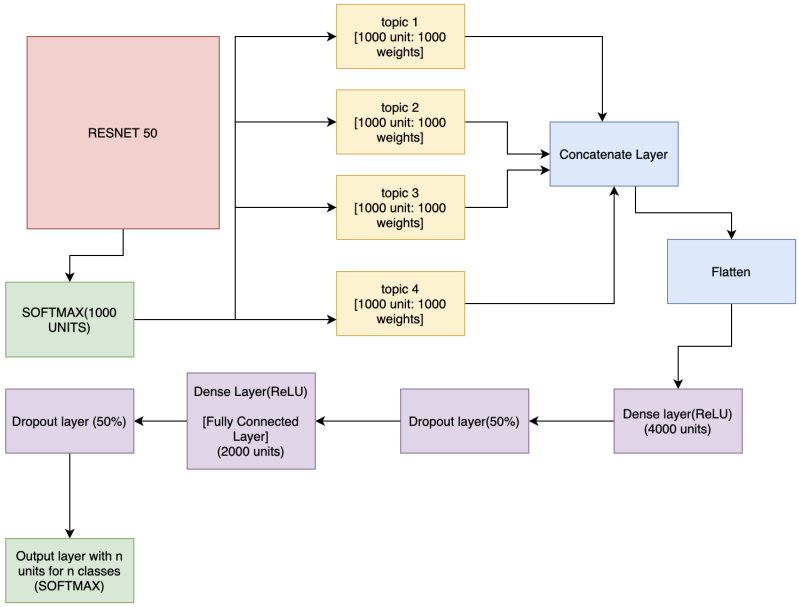


Figure 25: Topic Model Implementation 1 Flow Diagram

5.4.2 Implementation 2

The fifth implementation of the Topic Model builds upon the previous iterations and makes use of the skip connections provided by the Resnet network. This implementation takes inspiration from the use of Max Pooling layers from the previous implementation and extends the concept. In this model, instead of using one max pooling layer, there are two max pooling layers used. The first max pooling layer has a stride of 4, similar to the one done previously as well as a second max pooling layer with a stride of 2. The second max pooling layer helps to acquire 500 best classes from the 1000 unit Resnet-50 softmax layer. The outputs from both the max pooling layer as well as the output from the Topic custom layer are combined together through the use of an Add layer.

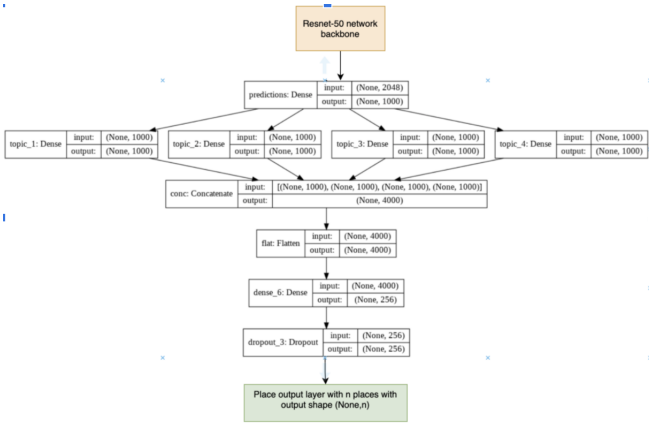


Figure 26: Topic Model Implementation 1 Architecture

This helps the model to learn from the top 250, 500 as well as the original 1000 objects. The disadvantage of this implementation is that it contains more parameters(11.24 Million). The model architecture for this implementation can be viewed in Figure 27.

6 Experiments

6.1 Validation Metrics Used

In order to evaluate the performance of the different model architectures introduced in the Proposed Methodology Section(Refer Sections 5.3 and 5.4), the following metrics are used :

- **Cross-Entropy Loss** : Cross-Entropy Loss can be defined as the loss measured when two probability distributions are compared for a random variable or a set of events. This loss function is the basis for evaluating the loss of the different model architectures and is discussed in detail in Section 3.3 of the report. The formula for estimating categorical cross-entropy is re-iterated below:

$$CE = -\sum_{i=0}^{n-1} y_i * log(\hat{y}_i) \tag{12}$$

  Here $\hat{y}_i$ is referred to as the predicted probability and y_i is referred to as the actual probability for n number of classes. Here the classes refer to the number of places in the image dataset ranging from values 0 to $n-1$.

- **Categorical Accuracy** : The Categorical Accuracy is a variant of the Accuracy metric which measures the percentage of predicted class labels(places for the scope of the project) which match the actual place labels for a batch of data. Thus, categorical accuracy can be defined as

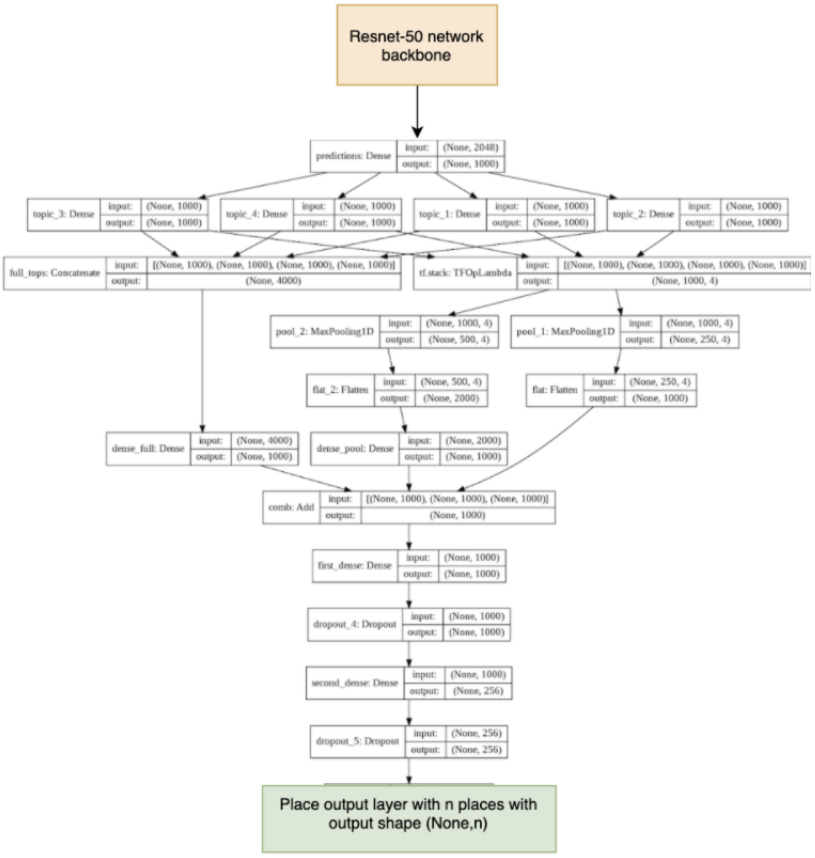


Figure 27: Topic Model Implementation 2 Architecture

the ratio of the correctly predicted places to the total number of observations in the dataset. The formula can be represented in the form of the following equation:

$$\text{Categorical Accuracy(CA)} = \frac{Count(\text{Correct Predictions})}{Count(\text{Total Observations})} \tag{13}$$

- **Precision** : Precision is sometimes also called the Positive Prediction Rate(PPR). It can be defined as the ratio of the number of True Positive instances(The relevant values which are positively identified as relevant) to the count of the total number of positive observations found in the model. The total number of positive observations made by the model can also include some false positive values(observations deemed relevant by the model which are not relevant). The formula for determining Precision

is given by the following equation:

$$\text{Precision} = \frac{Count(\text{True Positives})}{Count(\text{Positive Values})} \tag{14}$$

- **Recall** : Recall is sometimes also deemed as Sensitivity. Recall can be defined as the ratio of the count of correct positive predictions per class which are made by the model to the sum of the total number of true positives and false negatives made by the model. Unlike Precision, Recall also helps the end-user to observe the distribution for the False Negatives observations predicted by the model. False Negatives refer to the values which are predicted by the model as not-relevant to a class but are in reality relevant. The formula for Recall can be summarized by the following equation :

$$\text{Recall} = \frac{Count(\text{True Positives})}{Count(\text{True Positives}) + Count(\text{False Negatives})} \tag{15}$$

In the case of model prediction, the main parameter apart from Precision and Recall was the Support and the F-1 Score which are defined below :

- **Support** : Support is a metric which views the prevalence of a particular class of observation in the dataset. It is defined by the following equation:

$$\text{Support} = \frac{\text{Observation belonging to one class}}{\text{Total Observations}} \tag{16}$$

- **F-1 Score** : The F-1 score of a model is defined as the harmonic mean of the Precision and Recall of the model. It is also termed as the Sørensen–Dice coefficient. This metric pertains to the accuracy of the testing hypothesis. Thus this can be defined by the following equation :

$$F_1 = 2\frac{\text{precision.recall}}{\text{recall} + \text{precision}} = \frac{Count(\text{TP})}{Count(\text{TP}) + 0.5(Count(\text{FP}) + Count(\text{FN}))} \tag{17}$$

The last two metrics will be used in Section 7 where the model prediction performance will be assessed with the help of the Classification Report.

The metrics listed above can be readily implemented out of the box using Keras and TensorFlow libraries and are thus implemented from there. Additionally, Sigmoidal Focal Loss was also considered. The Sigmoidal Loss is an add-on package from TensorFlow 2.0 . This resulted in low training loss but resulted in high testing loss and lower prediction accuracy in the model. This was thus discarded and Categorical Cross-Entropy Loss was used throughout.

6.2 Stratified K-Fold Baseline Model for EgoPlaces

The train and val subsets for the EgoPlaces dataset are split into 4 folds using the Stratified K-Fold method. Each of the 4 folds of the dataset includes the

entirety of the 24 places. This dataset is then fed into the Baseline model for each fold. The performance of the model with the highest categorical accuracy and the lowest loss are chosen. Other metrics for evaluation are: Precision and Recall. The loss metric for the model is Cross-Entropy Loss. The hyper-parameters used for performing the model evaluation are provided in Table 6.

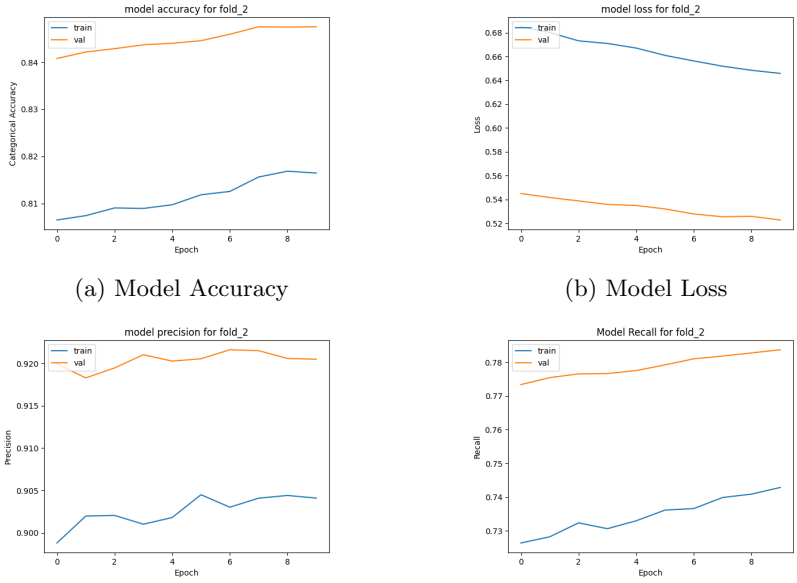
Parameters	**Value**
Number of Folds	4
Training Batch Size	128
Validation Batch Size	128
Learning Rate	0.01
Training Images Count	55227
Validation Images Count	18410
Number of epochs per fold	10
Count of places	24
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy, Precision, Recall

Table 6: Parameters used for running the Stratified K-Fold model for EgoPlaces dataset.

The values for the Train and Validation data for each fold can be referred from Table 7. From the results listed in Table 7 one can see that Fold number 2 gives the best model performance for the EgoPlaces dataset. The graphs for the metrics can be found in Figure 28.

	Metrics	**Fold 1**	**Fold 2**	**Fold 3**	**Fold 4**
Training Metrics	Loss	0.6256	0.6458	0.6303	0.6485
	Categorical Accuracy	81.98%	81.65%	81.90%	81.34%
	Precision	90.42%	90.41%	90.30%	90.31%
	Recall	75.05%	74.29%	74.76%	73.94%
Validation Metrics	Loss	0.5432	0.5228	0.5257	0.5500
	Categorical Accuracy	84.33%	84.75%	84.86%	83.83%
	Precision	91.39%	92.05%	91.90%	91.76%
	Recall	78.47%	78.37%	79.08%	77.70%

Table 7: Training and Validation metric values for each fold. The best-performing fold is highlighted in yellow.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

Figure 28: Graphs for Validation metrics for Fold 2

6.3 Baseline Model for EgoPlaces

6.3.1 Training the Baseline model with weights of Resnet-50 backbone model set as untrainable

The next experiment is performed for the Baseline Model on the EgoPlaces dataset. The Baseline model is fit with the training data and is trained for a set number of epochs. The model is validated during each epoch with the validation data. The trained model is then evaluated using the test dataset. The parameters for the experiment are defined in Table 8.

The Baseline model performance for EgoPlaces can be further assessed by referring to Table 9 which gives the final metric results for the training, validation and the testing dataset.

As one can observe from Table 9 and Figure 29. There is a potential bottleneck in the model. During the second run of the model, the accuracy of the model does not improve much. Thus we are required to fine-tune the model further in order to improve the performance of the model. This will be performed in the next step.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01 to 0.00001
Training Images Count	56495
Validation Images Count	7362
Validation Images Count	17142
Number of epochs	100
Count of Places	24
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy, Precision, Recall
Number of Trainable Parameters	265,368

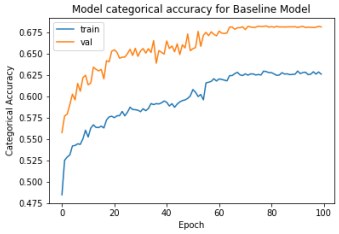
Table 8: Parameters used for running the Baseline model for EgoPlaces dataset.

Metrics	Train	Val	Test
Loss	1.3127	1.1884	1.0957
Categorical Accuracy	60.06%	65.35%	67.77%
Precision	83.65%	87.12%	86.79%
Recall	45.63%	48.29%	54.81%

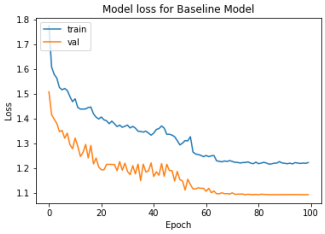
Table 9: Training and Validation metric values for Baseline Model using test, train and val dataset for 2 runs of 50 epochs each

6.3.2 Fine Tuning the Baseline Model for EgoPlaces

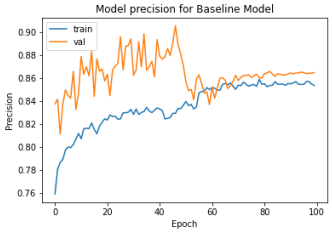
This experiment involves Fine-Tuning the model in order to eliminate the problems encountered in the previous experiment. In order to do this, the weights of the Baseline Model are loaded and the weights of the backbone Resnet-50 model are set to True. This is done so as to improve the model performance. When the model reaches 60% categorical accuracy, the performance of the model does not improve much. Thus by training the whole model again along with the backbone model, this effect can be reduced. The parameters used in order to fine-tune the model can be seen in Table 10. The metric values after Fine-Tuning for the different sub-sets of the data are summarized in Table 11 and the respective graphs for the metrics can be viewed in Figure 30.



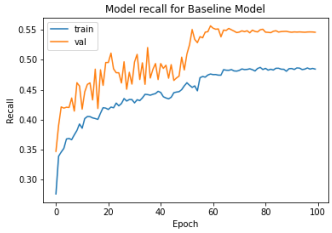
(a) Model Accuracy



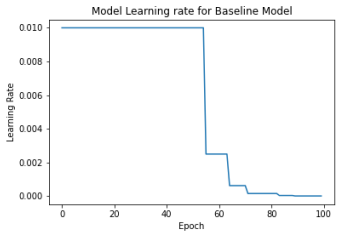
(b) Model Loss



(c) Model Precision



(d) Model Recall



(e) Model Learning Rate

Figure 29: Graphs for Validation metrics for Baseline Model and backbone model weights set to False

6.4 Baseline Model for Places 365

The Baseline Model with top layer containing 365 units is trained with the Places365 dataset. The model is trained for 35 epochs and is evaluated at the end of training.

The parameters for the experiment are summarized in Table 12. The values for the validation metrics for the train and val subset for Places-365 dataset can be viewed in Table 13. The graphs for the metrics can be found in Figure 31.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01
Training Images Count	56495
Validation Images Count	7362
Validation Images Count	17142
Number of epochs	30
Count of Places	24
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	23,799,960

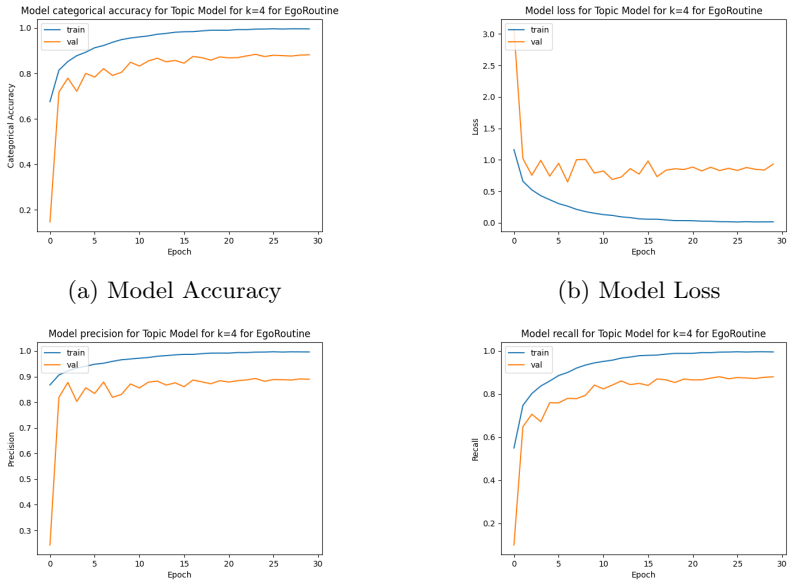
Table 10: Parameters used for fine-tuning the Baseline model for EgoPlaces dataset.

Metrics	Train	Val	Test
Loss	0.0154	0.9332	0.8928
Categorical Accuracy	99.56%	88.19%	87.94%
Precision	99.62%	88.95%	88.76%
Recall	99.50%	87.95%	87.69%

Table 11: Training and Validation metric values for Baseline Model after Fine-Tuning the model

Parameters	Value
Training Batch Size	128
Validation Batch Size	128
Learning Rate	0.45
Training Images Count	182500
Validation Images Count	36500
Number of epochs	35
Count of Places	365
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	2,472,301

Table 12: Parameters used for training the Baseline model for Places365 dataset.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

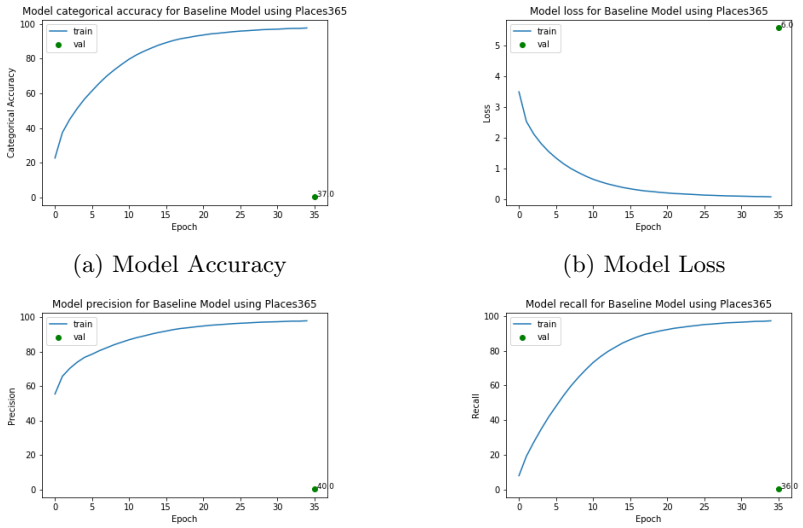
Figure 30: Graphs for Validation metrics for Baseline Model after Fine-Tuning

Metrics	**Train**	**Val**
Loss	0.0769	5.5793
Categorical Accuracy	97.62%	37.07%
Precision	97.96%	40.33%
Recall	97.28%	35.58%

Table 13: Training and Validation metric values for Baseline Model for Places365 dataset.

6.5 Topic Model Implementation 1 for EgoPlaces

After training and evaluating the Baseline models for both Places365 and EgoPlaces dataset, we train the first implementation of the Topic Model. This model is trained on the same dataset as the baseline and are then compared. The difference in the baseline model and the topic models are the dense layers. Each of these dense layers serve as respective topics with contain the 1000 objects from the soft-max layer of the backbone Resnet-50 network. The number of dense layers required depend on the number of topics which are to be determined. These topic model architectures are designed in a way such that each topic dense layer derives the 1000 objects from the Resnet-50 soft-max layer separately. These topic dense layers do not depend on each other and derive the objects separately from the Resnet-50 predictions layer. In the case of the



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

Figure 31: Graphs for Validation metrics for Baseline Model with Places 365 Dataset. The green point demarcates the values for the validation data

Baseline model, the objects are not considered and the images are classified into their respective Place labels. This implementation of the Topic model makes use of all the objects from the Resnet-50 softmax layer which are then classified into the respective place labels through the means of a Concatenation layer followed by a Flatten layer. The concatenate layer helps to coalesce the topic dense layers and the flatten layer takes the concatenated layer output and feeds the information to the Classification layer which in turn classifies the batches of images into their respective places.

The configuration of the experiments performed are summarized in Table 14 and the results of the model performance during training and evaluation can be found in 15.

The training and evaluation statistics for this implementation of the Topic Model can be found in Figure 32.

6.6 Topic Model Implementation 2 for EgoPlaces

The second Topic Model architecture is an extension of the first implementation of the Topic Model along with the concept of Pooling layers introduced in another variant of the Topic Model(Refer Appendix 11.1.3). This architecture, helps to classify a batches of images into their corresponding place labels by extracting the top 500 as well as the top 250 objects taken from the Resnet-50 soft-max layer. These two collections along with the output from the Resnet-50 soft-max layer are then sent to the classification layers for processing. This

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01-0.001
Training Images Count	56495
Validation Images Count	7362
Testing Images Count	17142
Number of epochs	200
Count of Places	24
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	5,030,424

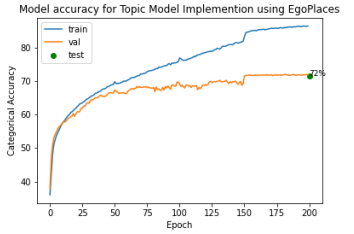
Table 14: Parameters used for training Topic model: Implementation 1 for EgoPlaces dataset.

Metrics	Train	Val	Test
Loss	0.4535	1.1472	1.1402
Categorical Accuracy	86.53%	71.92%	71.63%
Precision	94.51%	80.27%	80.24%
Recall	80.48%	66.34%	65.82%

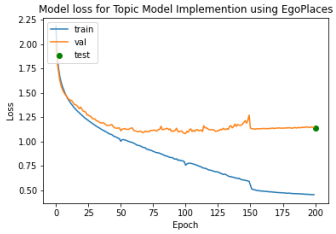
Table 15: Training and Validation metric values for Topic Model : Implementation 1 for EgoPlaces dataset.

helps the model to better learn the distribution of objects across the image in theory as it processes the top 500 best performing objects as well as the top 250 performing objects along with the original 1000 objects from the Resnet-50 soft-max layer. The output of these three pipelines are then processed using a concatenation layer followed by a flattening layer(Refer Figure 27 for the architecture). The output of these layers are connected to the classification layers. Due to the increased complexity of the model as compared to the first implementation, the number of parameters are significantly increased. However, as shown in Figure 33, this architecture results in better accuracy, precision and recall with a lower Cross-Entropy loss as compared to Implementation 1 as well the Baseline model for the EgoPlaces dataset for Training.

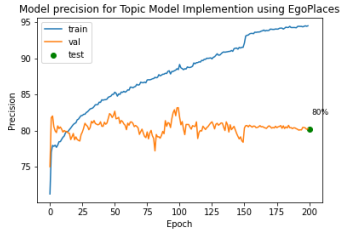
The configuration for training and evaluating this model architecture along with the Validation Metrics for train, test and validation can be seen in Tables 16 and 17 respectively.



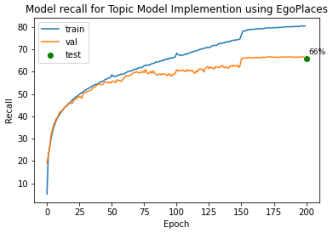
(a) Model Accuracy



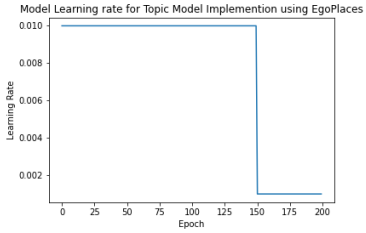
(b) Model Loss



(c) Model Precision



(d) Model Recall



(e) Model Learning Rate

Figure 32: Graphs for Validation metrics for Topic Model Implementation 1 using EgoPlaces dataset

6.7 Topic Model Implementation 1 for Places365

After Training the Topic Model on the EgoPlaces dataset, we now perform model training on the Places365 dataset. This dataset contains 365 place labels. Thus the final classification(top layer) of the model contains 365 neuron units for the same.

Thus this implementation of the Topic Model takes in 1000 objects from the backbone Resnet-50 architecture which is pre-trained on the ImageNet dataset. The output from this softmax layer is then fed as input to the four distinct Topic Dense Layers, each with 1000 units. The output from the different topic dense layers are then concatenated and flattened before passing them through the Classification layers. A few intermediate Dropout layers are adopted in order to reduce the chance of over-fitting. The model is trained on a subset of the Places365 dataset which includes 1500 images for the different 365 places.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01-0.001
Training Images Count	56495
Validation Images Count	7362
Testing Images Count	17142
Number of epochs	170
Count of Places	24
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	11,265,424

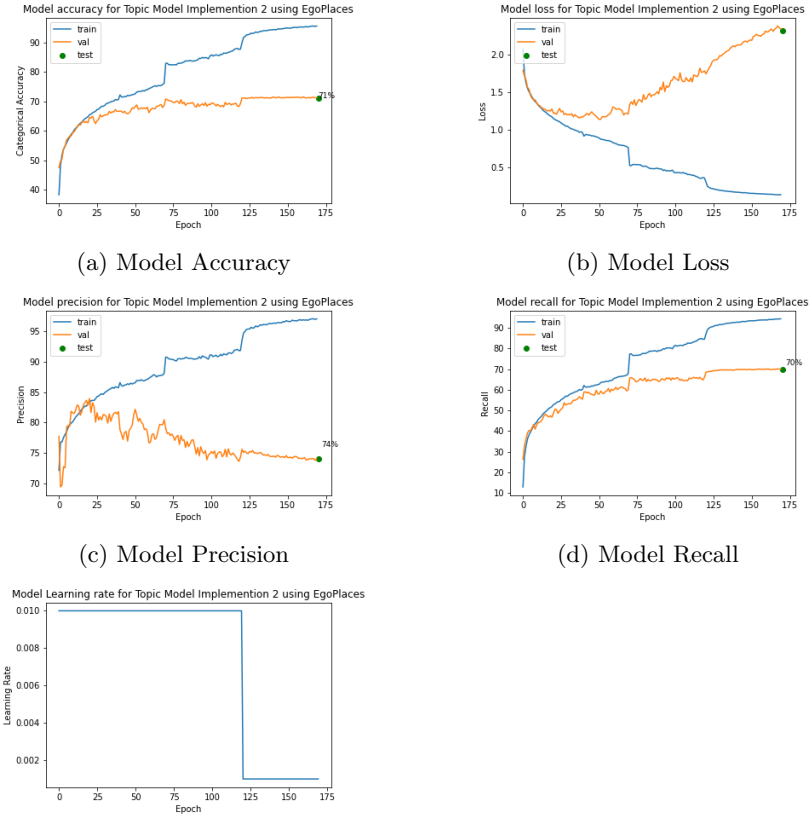
Table 16: Parameters used for training Topic model: Implementation 2 for EgoPlaces dataset.

Metrics	Train	Val	Test
Loss	0.1295	2.3382	2.3182
Categorical Accuracy	95.61%	71.24%	71.21%
Precision	97.08%	74.05%	74.05%
Recall	94.35%	69.98%	69.87%

Table 17: Training and Validation metric values for Topic Model : Implementation 2 for EgoPlaces dataset.

The parameters used for training the model are summarized in Table 18. The model is then evaluated using the 36500 validation images with 100 images for each place label. The results of the training and evaluation of the model are summarized in Table 19.

The graphs for the final results for training and evaluation can be found in Figure 34.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

(e) Model Learning Rate

Figure 33: Graphs for Validation metrics for Topic Model Implementation 2 using EgoPlaces dataset

6.8 Topic Model Implementation 2 for Places365

The second implementation of the Topic Model is trained using the Places365 dataset. This implementation of the Topic Model is facilitated by max pooling the 500 best objects as well as the 250 best objects identified for the different place labels provided by the dataset along with the 1000 objects all of which are connected together with the help of skip connections. The output from these three branches in the model architecture are then connected using a concatenation layer followed by a flattening layer(Refer the architecture in Figure 27). Some intermediate dropout layers are included in the model in order to prevent it from over-fitting.

The parameters for the model training and testing are summarized in Table 20. The values for the metrics in Training and Testing are illustrated in Table 21.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Learning Rate	0.01
Training Images Count	182500
Testing Images Count	36500
Number of epochs	330
Count of Places	365
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	8,366,365

Table 18: Parameters used for training Topic model: Implementation 1 for Places365 dataset.

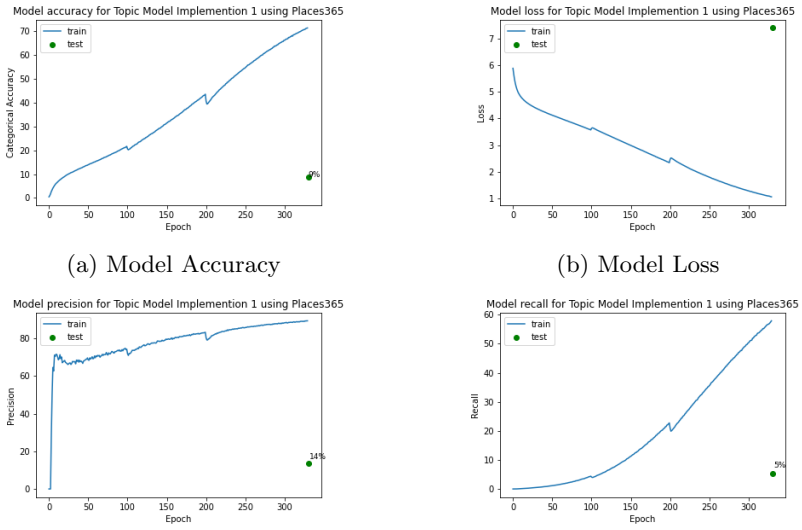
Metrics	Train	Test
Loss	1.0673	7.3914
Categorical Accuracy	71.45%	8.62%
Precision	89.38%	13.82%
Recall	57.90%	5.45%

Table 19: Training and Validation metric values for Topic Model : Implementation 1 for Places365 dataset.

The graphs for the different validation metrics for the model performance during training and testing can be seen from Figure 35.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Learning Rate	0.01-0.003
Training Images Count	182500
Testing Images Count	36500
Number of epochs	210
Count of Places	365
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	11,702,757

Table 20: Parameters used for training Topic model: Implementation 2 for Places365 dataset.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

Figure 34: Graphs for Validation metrics for Topic Model Implementation 1 using Places365 dataset

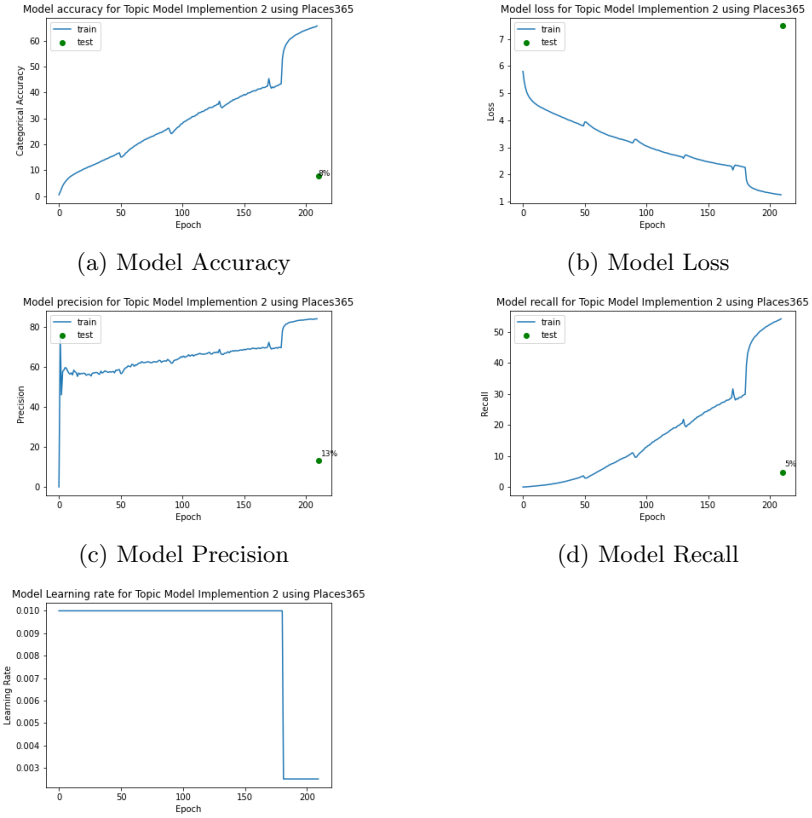
Metrics	Train	Test
Loss	1.2553	7.4770
Categorical Accuracy	65.67%	7.77%
Precision	83.93%	13.26%
Recall	54.21%	4.60%

Table 21: Training and Validation metric values for Topic Model : Implementation 2 for Places365 dataset.

6.9 Creating the Places20 dataset

When assessing the model performance on the EgoPlaces and Places20 dataset, we make the following observations :

- The EgoPlaces dataset gives better accuracy for the Baseline and the Topic models.
- The EgoPlaces dataset contains ambiguity in the place labels as some of the images for a particular place do not represent the place clearly.
- The Places365 dataset has images much more representative of the place categories they belong too.
- However, the Places365 dataset contains a lot of classes to choose from and may result in poor model performance.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

(e) Model Learning Rate

Figure 35: Graphs for Validation metrics for Topic Model Implementation 2 using Places365 dataset

In order to address these issues, we create a new dataset. This dataset is a subset of the greater Places365[39]. However, only 20 place categories are in this subset of the dataset. For each place label for this dataset, we consider 1400 images for training with 100 images for validation and 100 images per place for performing predictions on the models. We also train the models with this dataset and perform experiments on the different model architectures.

The 20 places chosen for this so called Places20 dataset, we have the 20 places which are represented in Figure 36.

6.10 Baseline Model for Places20

In order to check the model performance on the newly created Places20 dataset, we first train the baseline model using the Places20 dataset. The first phase

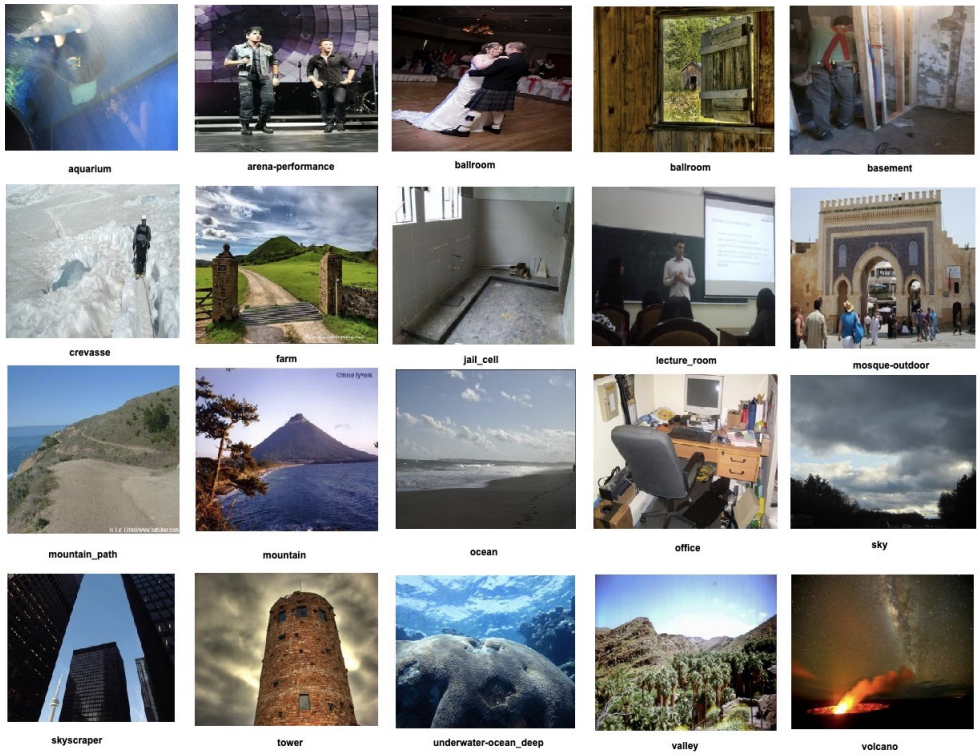


Figure 36: The 20 place categories for the Places20 dataset[39]

of the model training involves training the model without setting the backbone model as trainable and train the model till there is a bottleneck in the model performance. This is referred to the point where the model has better testing performance than the training performance. We then fine-tune the model as performance for the previous datasets and retrain the model and observe our results. The parameters used for training the model can be viewed in Table 22 and the model results can be seen in Table 23. The graphs for the model during model training and evaluation can be found in Figure 37.

6.11 Topic Model Implementation 1 for Places20

The baseline model gives a testing accuracy of 75%. We now train the Topic models for the Places20 dataset starting with the first implementation of the Topic Model.

The parameters for the model training are referred in Table 24. The results of the model training and validation can be found in Table 25. The graphs for the model training and testing performance can be found in Figure 39.

Parameters	Value
Training Batch Size	50
Validation Batch Size	50
Learning Rate	0.1
Training Images Count	28000
Validation Images Count	2000
Number of epochs	60
Count of Places	20
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	529,684 (no fine tuning) 24,064,276 (fine tuning)

Table 22: Parameters used for training the Baseline model for Places20 dataset

Metrics	Train	Val
Loss	0.0711	1.8019
Categorical Accuracy	97.84%	75.35%
Precision	98%	75.13%
Recall	97.62%	74.8%

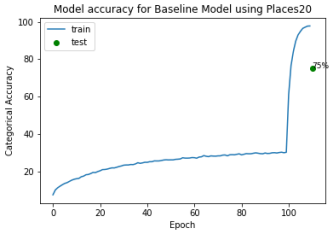
Table 23: Parameters used for training Baseline Model for Places20 dataset.

Parameters	Value
Training Batch Size	50
Validation Batch Size	50
Learning Rate	0.1
Training Images Count	28000
Validation Images Count	2000
Number of epochs	200
Count of Places	20
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	6,058,772

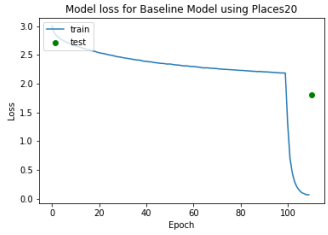
Table 24: Parameters used for training the Topic model implementation 1 for Places20 dataset

6.12 Topic Model Implementation 2 for Places20

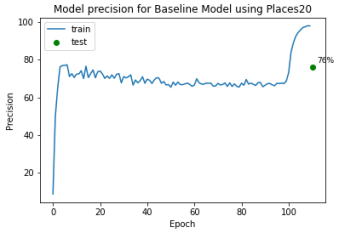
The Topic Model Implementation 2 through the max-pooling parts of the model can help the model learn the top 500 as well as the top 250 objects for a model along with the 1000 ImageNet objects obtained using the backbone Resnet-50 network. The parameters used for training and evaluating this model archi-



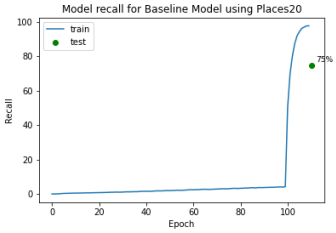
(a) Model Accuracy



(b) Model Loss



(c) Model Precision



(d) Model Recall

Figure 37: Graphs for Validation metrics for Baseline Model using Places20 dataset

Metrics	Train	Val
Loss	0.4509	3.8962
Categorical Accuracy	84.8%	36.85%
Precision	92.3%	44.08%
Recall	78.97%	33.90%

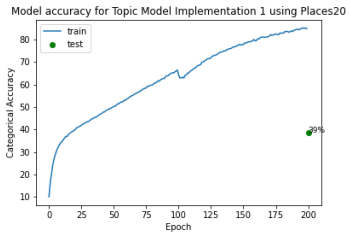
Table 25: Parameters used for training Topic Model Implementation 1 for Places20 dataset.

tecture are summarized in Table 26. The metric results during training and validating the model can be found in Table 27. The graphs for the model during training and validating the model are found in Figure 39.

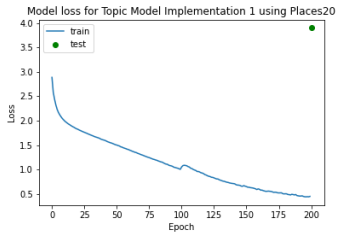
7 Results and Discussion

In this section, we summarize the results from the experiments in the previous section and make predictions on the Baseline model as well as the two Topic Models for both the datasets. This will help us to view the model performance on data which has not yet been processed by the different models.

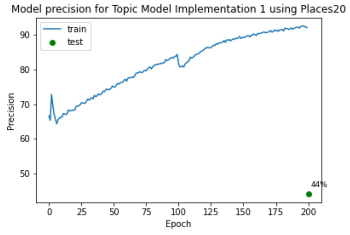
The first part of this section deals with the Prediction scores for the different model architectures, starting from the Baseline model and followed by



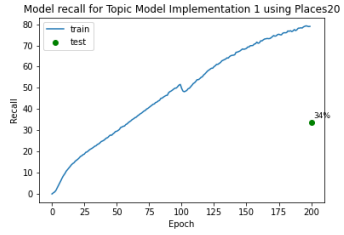
(a) Model Accuracy



(b) Model Loss



(c) Model Precision



(d) Model Recall

Figure 38: Graphs for Validation metrics for Topic Model Implementation 1 using Places20 dataset

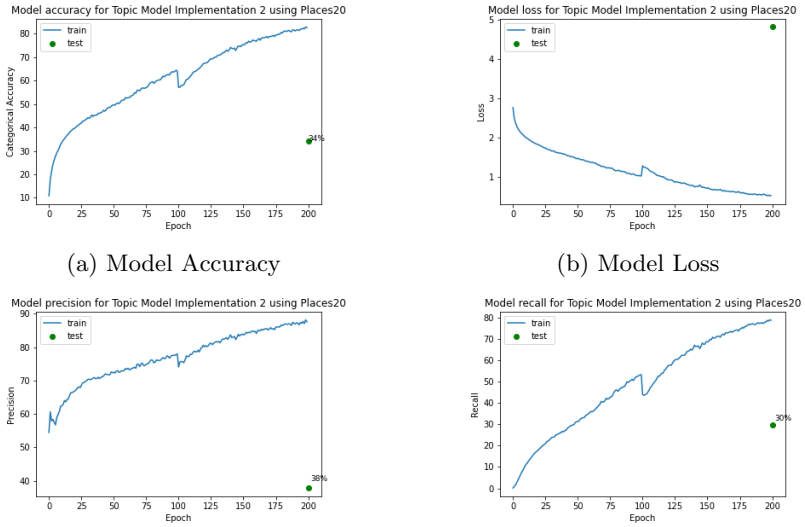
Parameters	**Value**
Training Batch Size	50
Validation Batch Size	50
Learning Rate	0.1
Training Images Count	28000
Validation Images Count	2000
Number of epochs	200
Count of Places	20
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	11,525,772

Table 26: Parameters used for training the Topic model implementation 2 for Places20 dataset

the different model architectures of the Topic Models. The prediction performance of the models are measured using the Confusion Matrix as well as the Classification Report of the prediction results. The Classification report is a tool provided by the SciPy module in Python which helps assessing the Prediction, Recall as well as the F1-Score of the model for each place label. The report also provides the amount of support for each place label. This is defined

Metrics	Train	Val
Loss	0.5317	4.8230
Categorical Accuracy	82.61%	34.2%
Precision	87.56%	37.82%
Recall	78.67%	29.90%

Table 27: Parameters used for training Topic Model Implementation 2 for Places20 dataset.



(a) Model Accuracy

(b) Model Loss

(c) Model Precision

(d) Model Recall

Figure 39: Graphs for Validation metrics for Topic Model Implementation 2 using Places20 dataset

as the amount of images belonging to a particular place label in the test dataset.

This will be performed for EgoPlaces as well as the Places365 dataset.

The second part of this section pertains to exploring the objects visualized by the model in detail in order to better understand the functioning of the Topic Models. In order to perform this, we use the best model implementation in order to derive our results. The performance of some other model architectures used in the project can be viewed in the Appendix section of the report(Appendix Section 11.1).

7.1 Performing model predictions on EgoPlaces dataset

7.1.1 Baseline Model

The Baseline Model performance is evaluated using the testing dataset from EgoPlaces. The images in this subset of the dataset are not used for training and validating the model. The predictions using this model are assessed using a Confusion Matrix followed by a Classification Report.

Confusion Matrix : The Confusion matrix for the Baseline Model can be viewed in Figure 40.

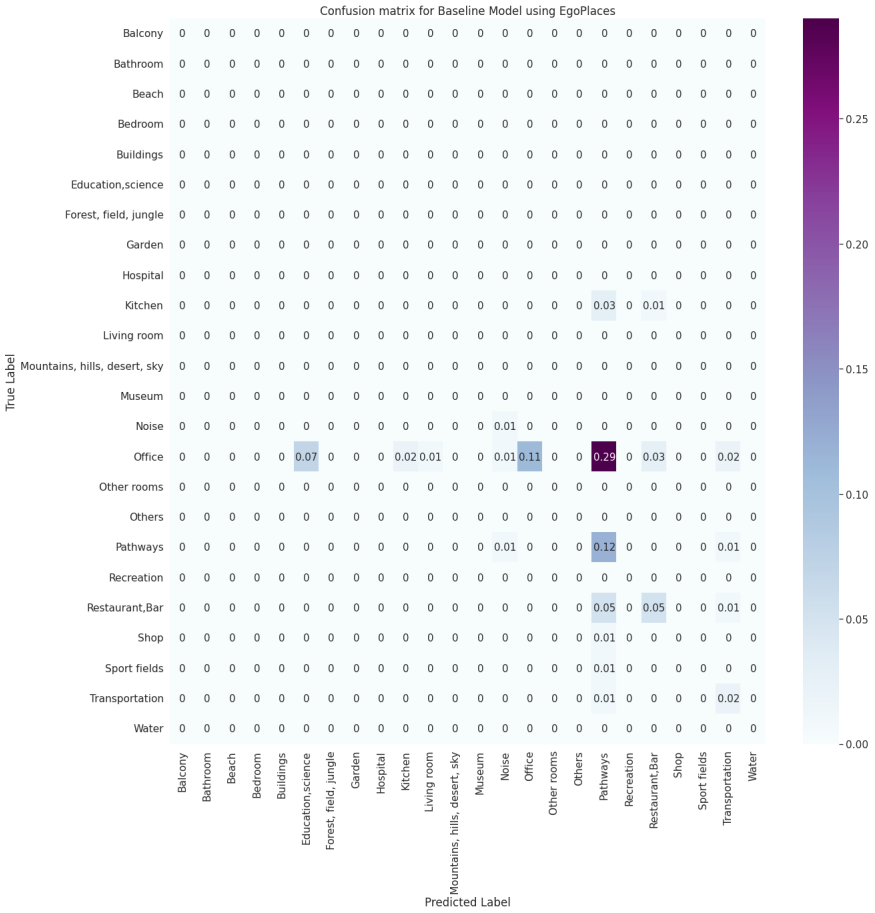


Figure 40: Baseline Model Prediction Confusion Matrix for EgoPlaces dataset using new data

Classification Report : The Classification Report for the Baseline Model performance can be viewed in Figure 41.

From the classification report in Figure 41, we can observe that the baseline model is able to identify about 5 places from the test data set at a precision of over 25%. The overall accuracy of the model prediction turns out to be around 31%.

	precision	recall	f1-score	support
Balcony	0.00	0.00	0.00	30
Bathroom	0.71	0.04	0.07	141
Beach	0.00	0.00	0.00	1
Bedroom	0.00	0.00	0.00	55
Buildings	0.00	0.00	0.00	39
Education,science	0.02	0.22	0.03	90
Forest, field, jungle	0.00	0.00	0.00	7
Garden	0.00	0.00	0.00	4
Hospital	0.00	0.00	0.00	11
Kitchen	0.09	0.03	0.05	869
Living room	0.02	0.12	0.04	58
Mountains, hills, desert, sky	0.80	0.03	0.06	121
Museum	0.00	0.00	0.00	56
Noise	0.16	0.67	0.26	135
Office	0.99	0.19	0.31	9007
Other rooms	0.01	0.02	0.01	60
Others	0.00	0.00	0.00	3
Pathways	0.22	0.82	0.35	2260
Recreation	0.00	0.00	0.00	87
Restaurant,Bar	0.48	0.43	0.45	1862
Shop	0.12	0.01	0.02	238
Sport fields	1.00	0.01	0.02	108
Transportation	0.25	0.64	0.35	451
Water	0.00	0.00	0.00	2
accuracy			0.31	15695
macro avg	0.20	0.13	0.08	15695
weighted avg	0.69	0.31	0.30	15695

Figure 41: Classification Report for Baseline Model for EgoPlaces dataset using new data

7.1.2 Topic Model : Implementation 1

Once the Baseline models are predicted on the test data, the same is performed on the different Topic Models proposed in the project starting from Implementation 1.

Confusion Matrix : The Confusion matrix for the first implementation of the Topic Model can be viewed in Figure 42.

Classification Report : The Classification Report for the Topic Model Implementation 1 performance can be viewed in Figure 43. From Figure 43, we can observe the model performance for the Topic Model Implementation 1. This

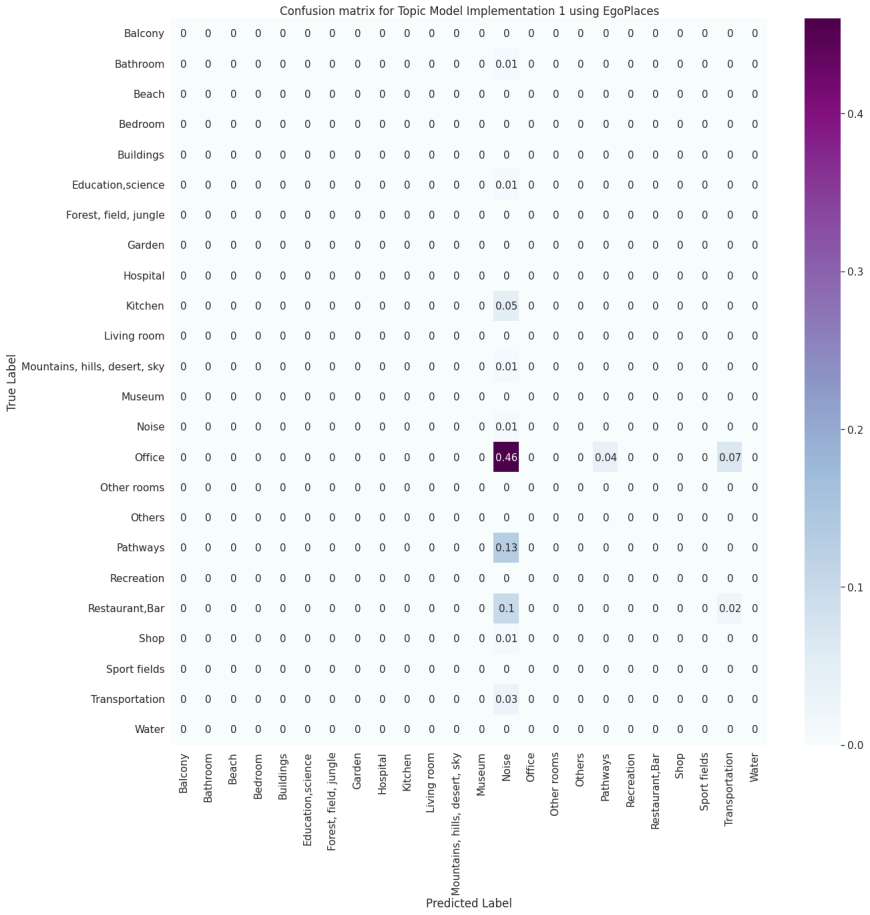
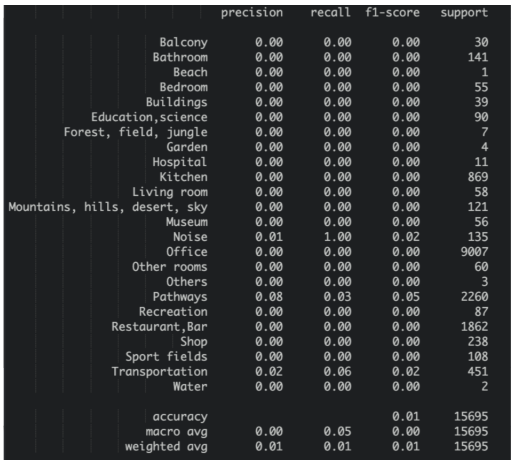


Figure 42: Topic Model Implementation 1 Prediction Confusion Matrix for EgoPlaces dataset using new data

model is able to identify 2 places when predicted on the test dataset. However, the model performance is heavily hindered by the Noise in the dataset presented by the Noise place label in the test data. The prediction accuracy of the model comes to about 1%. This is substantially lower than the Baseline results for the same dataset.

7.1.3 Topic Model : Implementation 2

Following the model prediction process on implementation 1 of the Topic Model, we perform the same operation on the second implementation. This implementation had better results on model training and evaluation. So in theory, this should perform better than the previous implementation in theory. The Confusion Matrix and the Classification report are once again plotted for the test



	precision	recall	f1-score	support
Balcony	0.00	0.00	0.00	30
Bathroom	0.00	0.00	0.00	141
Beach	0.00	0.00	0.00	1
Bedroom	0.00	0.00	0.00	55
Buildings	0.00	0.00	0.00	39
Education,science	0.00	0.00	0.00	90
Forest, field, jungle	0.00	0.00	0.00	7
Garden	0.00	0.00	0.00	4
Hospital	0.00	0.00	0.00	11
Kitchen	0.00	0.00	0.00	869
Living room	0.00	0.00	0.00	58
Mountains, hills, desert, sky	0.00	0.00	0.00	121
Museum	0.00	0.00	0.00	56
Noise	0.01	1.00	0.02	135
Office	0.00	0.00	0.00	9007
Other rooms	0.00	0.00	0.00	60
Others	0.00	0.00	0.00	3
Pathways	0.08	0.03	0.05	2260
Recreation	0.00	0.00	0.00	87
Restaurant,Bar	0.00	0.00	0.00	1862
Shop	0.00	0.00	0.00	238
Sport fields	0.00	0.00	0.00	108
Transportation	0.02	0.06	0.02	451
Water	0.00	0.00	0.00	2
accuracy			0.01	15695
macro avg	0.00	0.05	0.00	15695
weighted avg	0.01	0.01	0.01	15695

Figure 43: Classification Report for Topic Model Implementation 1 for EgoPlaces dataset using new data

data and the results can be viewed below.

Confusion Matrix : The Confusion matrix for the second implementation of the Topic Model can be viewed in Figure 44.

Classification Report : The Classification Report for the Topic Model Implementation 2 performance can be viewed in Figure 45.

As shown in Figure 45, the Implementation 2 Topic Model fares somewhat better than the first implementation. It is able to identify two places with high precision. However, similar to the first implementation, this model is also susceptible to the images belonging to the Noise class. The most predicted place label by this model is the Noise label for a majority of images belonging to the $Class$ label. Thus one can hypothesize that the model performance for the Topic Models could be better if the images with class $Noise$ are eliminated from the dataset.

Another observation can be made around the distribution and the amount of images belonging to a single place label. It can be seen that the dataset is quite imbalanced with the majority of images belonging to the place label $Office$ with other places having much fewer examples in comparison.

7.2 Performing model predictions on Places365 dataset

After performing predictions on the EgoPlaces dataset, we now perform the predictions on the Places365 dataset. In order to evaluate the prediction performance for the different model architectures, we make use of the same two methodologies namely, deriving the Confusion Matrix as well as the Classification Report. The test dataset for Places365 dataset was derived from the small

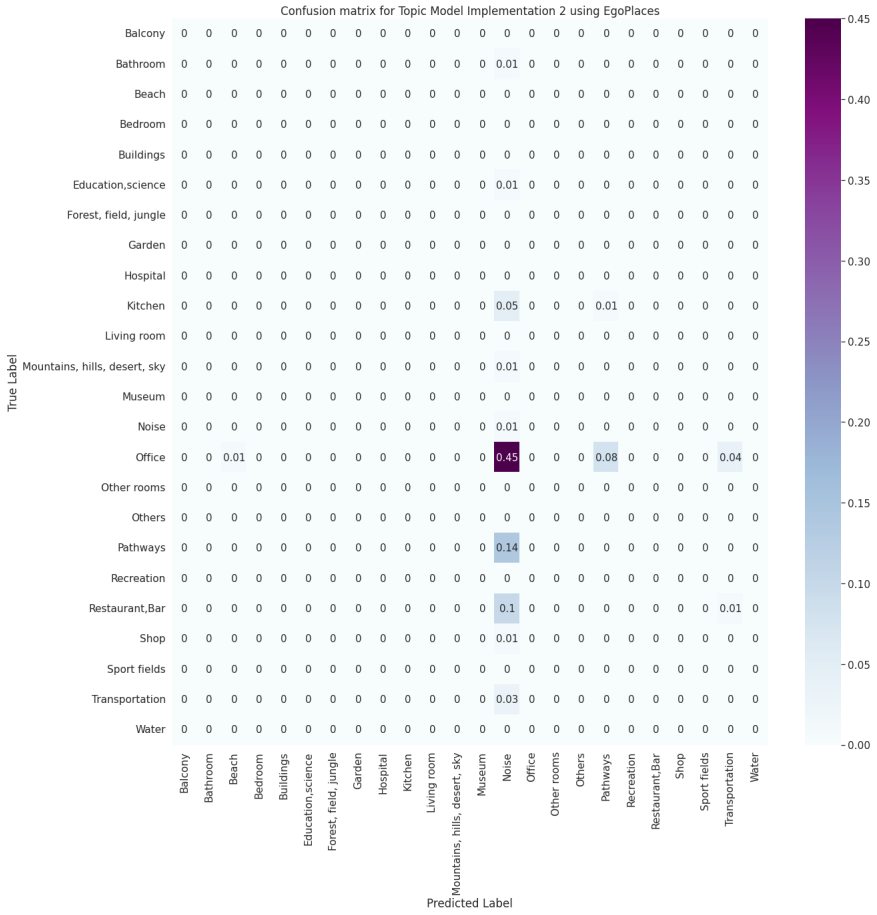


Figure 44: Topic Model Implementation 2 Prediction Confusion Matrix for EgoPlaces dataset using new data

Places365 dataset with 100 images present in each place class such that each image is not present in the training as well as the dataset used for validation.

7.2.1 Baseline Model

We first predict the Places365 test dataset introduced previously using the Baseline Model. The Baseline model is trained from scratch using the training dataset. We then plot the Confusion Matrix as well as procure the classification report for the same.

Confusion Matrix : The Confusion matrix for the second implementation of the Topic Model can be viewed in Figure 44.

	precision	recall	f1-score	support
Balcony	0.00	0.00	0.00	30
Bathroom	0.00	0.00	0.00	141
Beach	0.00	0.00	0.00	1
Bedroom	0.00	0.00	0.00	55
Buildings	0.00	0.00	0.00	39
Education,science	0.00	0.00	0.00	90
Forest, field, jungle	0.00	0.00	0.00	7
Garden	0.00	0.00	0.00	4
Hospital	0.00	0.00	0.00	11
Kitchen	0.00	0.00	0.00	869
Living room	0.00	0.00	0.00	58
Mountains, hills, desert, sky	0.00	0.00	0.00	121
Museum	0.00	0.00	0.00	56
Noise	0.01	1.00	0.02	135
Office	1.00	0.00	0.00	9007
Other rooms	0.00	0.00	0.00	60
Others	0.00	0.00	0.00	3
Pathways	0.05	0.03	0.04	2260
Recreation	1.00	0.01	0.02	87
Restaurant,Bar	0.00	0.00	0.00	1862
Shop	0.00	0.00	0.00	238
Sport fields	0.00	0.00	0.00	108
Transportation	0.02	0.03	0.02	451
Water	0.00	0.00	0.00	2
accuracy			0.01	15695
macro avg	0.09	0.04	0.00	15695
weighted avg	0.59	0.01	0.01	15695

Figure 45: Classification Report for Topic Model Implementation 1 for Ego-Places dataset using new data

Classification Report : The Classification Report for the Baseline Model performance can be viewed in Figure 28. This report contains the places having the highest F1 score from the dataset. The entire classification report can be viewed in detail in the Appendix section of the report(Table 45 in Appendix 11.5).

As we are evaluating the performance of the model on all 365 place labels from the Places365 dataset, the confusion matrix as seen in Figure 46 is challenging to read due to its size. Thus we will only use the Classification Report in order to estimate the model performance. The confusion matrix shows the positive values in the case of places which have the same value on prediction as their actual value. This value ranges from 0.00025 to 0.002 . The accuracy of the model prediction is around 37%.

7.2.2 Topic Model : Implementation 1

Once the Baseline Model has been predicted using the Places365 dataset, one can try to predict the test dataset for Places365 using the first implementation of the Topic Model. Due to sheer amount of place labels present in the dataset, plotting a confusion matrix would not be a good idea for this dataset as it may be too convoluted to see the results in detail. Thus the only metric to assess the model performance would be to create a Classification Report.

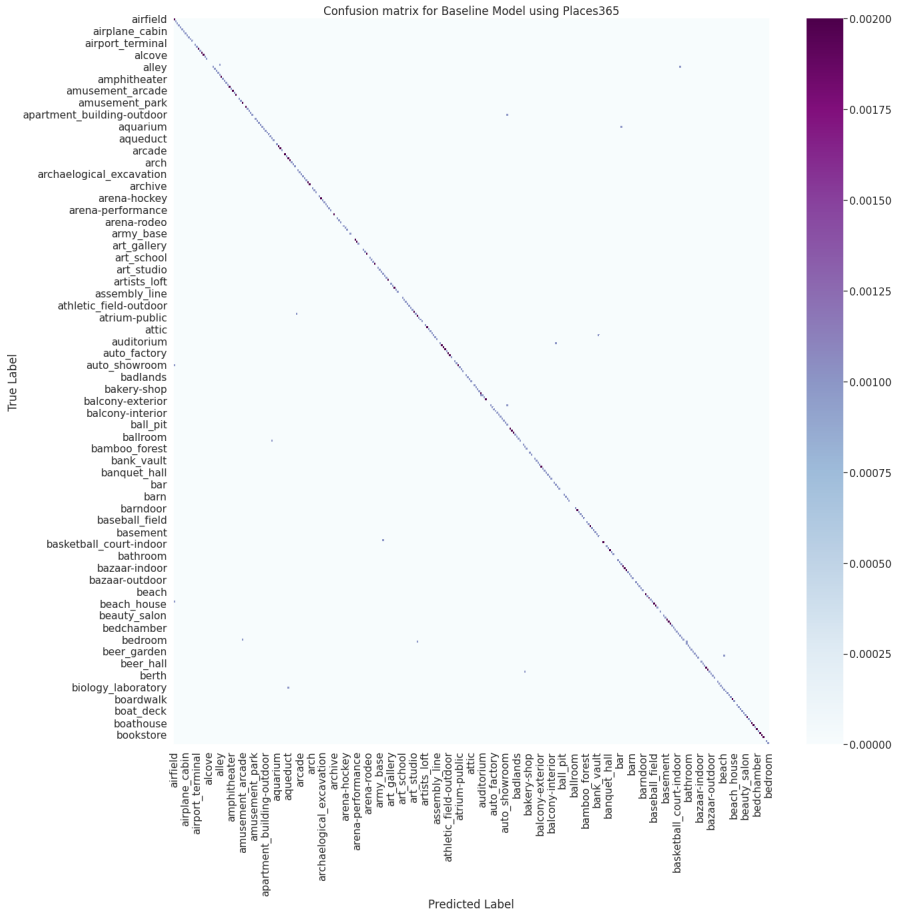


Figure 46: Baseline Model Prediction Confusion Matrix using new data on Places365 dataset

Classification Report : The classification report for the Topic Model Implementation 1 can be found in Table 29.

7.2.3 Topic Model : Implementation 2

After assessing the model prediction for the first Topic Model, the same can be performed for the second model implementation. As previously stated, the confusion matrix for classification of 365 places would be very convoluted. Thus classification report will be the only parameter which will help in order to check the model performance on previously unseen data.

Classification Report : The classification report for the Implementation 2 of the Topic Model can be found in Table 30.

Place	Precision	Recall	F1-score	Support
car_interior	0.752	0.880	0.811	100.000
ball_pit	0.857	0.720	0.783	100.000
cockpit	0.743	0.810	0.775	100.000
gymnasium-indoor	0.738	0.760	0.749	100.000
army_base	0.700	0.770	0.733	100.000
bus_station-indoor	0.823	0.650	0.726	100.000
boxing_ring	0.727	0.720	0.724	100.000
basketball_court-indoor	0.624	0.830	0.712	100.000
wind_farm	0.734	0.690	0.711	100.000
underwater-ocean_deep	0.750	0.660	0.702	100.000
carrousel	0.879	0.580	0.699	100.000
pizzeria	0.759	0.630	0.689	100.000
windmill	0.643	0.720	0.679	100.000
water_tower	0.667	0.660	0.663	100.000
phone_booth	0.713	0.620	0.663	100.000
accuracy	0.375	0.375	0.375	0.375
macro avg	0.388	0.375	0.372	36500.000
weighted avg	0.388	0.375	0.372	36500.000

Table 28: Top 10 Place label predictions for Baseline Model with highest F1-Score

Precision	Precision	Recall	F1-Score	Support
wind_farm	0.014	0.300	0.027	100.000
desert-sand	0.013	0.310	0.024	100.000
elevator-door	0.013	0.160	0.024	100.000
bow_window-indoor	0.500	0.010	0.020	100.000
skyscraper	0.250	0.010	0.019	100.000
attic	0.167	0.010	0.019	100.000
corridor	0.012	0.030	0.017	100.000
berth	0.045	0.010	0.016	100.000
elevator_shaft	0.045	0.010	0.016	100.000
hospital	0.034	0.010	0.016	100.000
accuracy	0.005	0.005	0.005	0.005
macro avg	0.003	0.005	0.001	36500.000
weighted avg	0.003	0.005	0.001	36500.000

Table 29: Classification Report for Topic Model Implementation 1 with top 10 places with the highest F1-Score for Places365 dataset

The overall accuracy for the topic model is quite low at around 0.4% . The model is able to identify around 15 places. The one with the highest F1-Score being *Ocean*. The model is able to predict the class *jail_cell* at a precision of 1

Place	Precision	Recall	F1-score	Support
ocean	0.016	0.030	0.021	100.000
jail_cell	1.000	0.010	0.020	100.000
home_theater	0.500	0.010	0.020	100.000
barndoor	0.333	0.010	0.019	100.000
tower	0.143	0.010	0.019	100.000
arena-performance	0.125	0.010	0.019	100.000
volcano	0.009	0.320	0.018	100.000
crevasse	0.010	0.020	0.013	100.000
skyscraper	0.008	0.020	0.011	100.000
mountain_path	0.013	0.010	0.011	100.000
accuracy	0.004	0.004	0.004	0.004
macro avg	0.006	0.004	0.001	36500.000
weighted avg	0.006	0.004	0.001	36500.000

Table 30: Classification Report for Topic Model Implementation 2 with top 10 places with the highest F1-Score for Places365 dataset

followed by *home_theater* at a precision of about 0.5 . The classification report for this model can be seen in detail in the Appendix section(Refer Table 46 in Appendix 11.5.2). A reason for the low accuracy of the model could be the extensive number of place labels that the model tries to classify the images in. The accuracy of the model may improve if more images are used for prediction.

7.3 Performing model predictions on Places20 dataset

After performing the experiments on the different model architectures using the Places20 dataset, the next step is performing the model predictions on these models to see whether they perform better than the one on the Places365 dataset.

Predictions are first performed on the Baseline model. The performance is then compared to the Topic Models using this dataset.

7.3.1 Baseline Model

The Baseline model implementation is used for performing predictions using the 20 place categories present in the Places20 dataset in order to check whether the model is able to perform better or comparable to the greater Places365 dataset. This is assessed by viewing the Confusion Matrix and the Classification Report for the same.

Confusion Matrix : The Confusion matrix for the Baseline Model can be viewed in Figure 47.

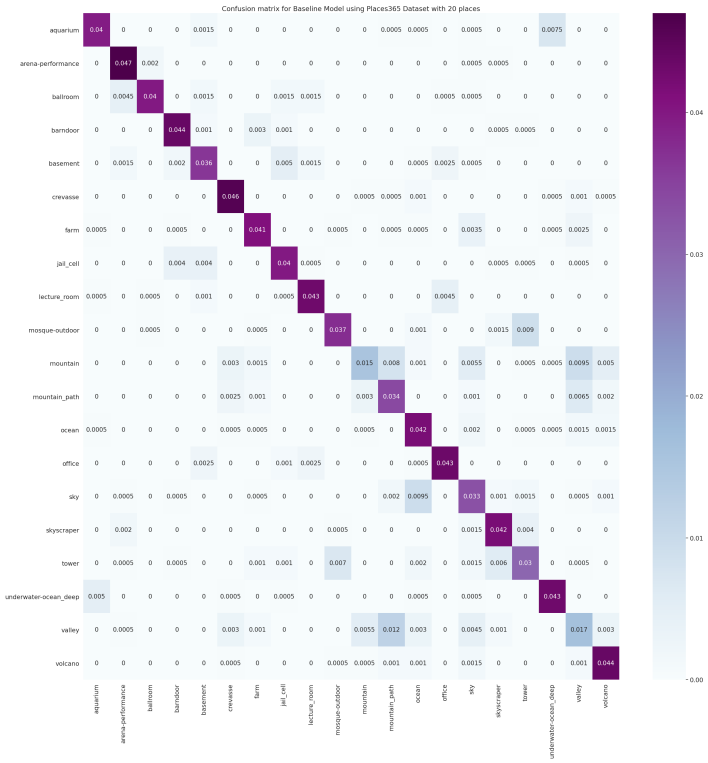


Figure 47: Baseline Model Prediction Confusion Matrix using new data on Places20 dataset

Classification Report : The Classification Report for the Baseline Model performance can be viewed in Table 31.

The baseline model is able to perform correct prediction for 18 out of the 20 place categories with high top-1 accuracy. The overall top-1 accuracy of the model stands at 76% . Thus the baseline model using the Places20 dataset has the best performance when compared to the same with EgoPlaces and Places365 dataset.

7.3.2 Topic Model Implementation 1

Once the predictions are performed on the baseline model, the next step is to perform the place prediction using the Topic Model Implementation 1 for the same data used as in the case of the baseline. We then create a Confusion Matrix for this as well as a Classification Report in order to perform the model performance comparison between this and the baseline model.

Confusion Matrix : The Confusion matrix for the Topic Model Implementation 1 can be viewed in Figure 48.

Places	precision	recall	f1-score	support
aquarium	0.86	0.79	0.82	100.00
arena-performance	0.83	0.94	0.88	100.00
ballroom	0.93	0.80	0.86	100.00
barndoor	0.85	0.88	0.87	100.00
basement	0.76	0.73	0.74	100.00
crevasse	0.82	0.92	0.87	100.00
farm	0.82	0.82	0.82	100.00
jail_cell	0.79	0.80	0.80	100.00
lecture_room	0.88	0.86	0.87	100.00
mosque-outdoor	0.82	0.75	0.78	100.00
mountain	0.61	0.31	0.41	100.00
mountain_path	0.58	0.68	0.63	100.00
ocean	0.67	0.84	0.74	100.00
office	0.85	0.87	0.86	100.00
sky	0.58	0.66	0.62	100.00
skyscraper	0.79	0.84	0.82	100.00
tower	0.65	0.60	0.62	100.00
underwater-ocean_deep	0.82	0.86	0.84	100.00
valley	0.41	0.33	0.37	100.00
volcano	0.77	0.88	0.82	100.00
accuracy				0.76
macro avg	0.75	0.76	0.75	2,000.00
weighted avg	0.75	0.76	0.75	2,000.00

Table 31: Classification Report for the Baseline Model for Places20 dataset

Classification Report : The Classification Report for the Baseline Model performance can be viewed in Figure 32.

A majority of the images from the test dataset get classified as sky. The place category sky is classified with a probability of 3% . The place : underwater-ocean_deep shows positive prediction accuracy at about 1%. The overall top-1 prediction accuracy of the model stands at about 8%.

7.3.3 Topic Model Implementation 2

The second implementation of the Topic Model is now used for predicting the places for the Places20 dataset. We again use Classification Report and Confusion Matrix in order to draw conclusions about the predictions. This will help to see how this model performance as compared to the Topic Implementation 1 as well as the Baseline Model.

Confusion Matrix : The Confusion matrix for the Topic Model Implementation 2 can be viewed in Figure 49.

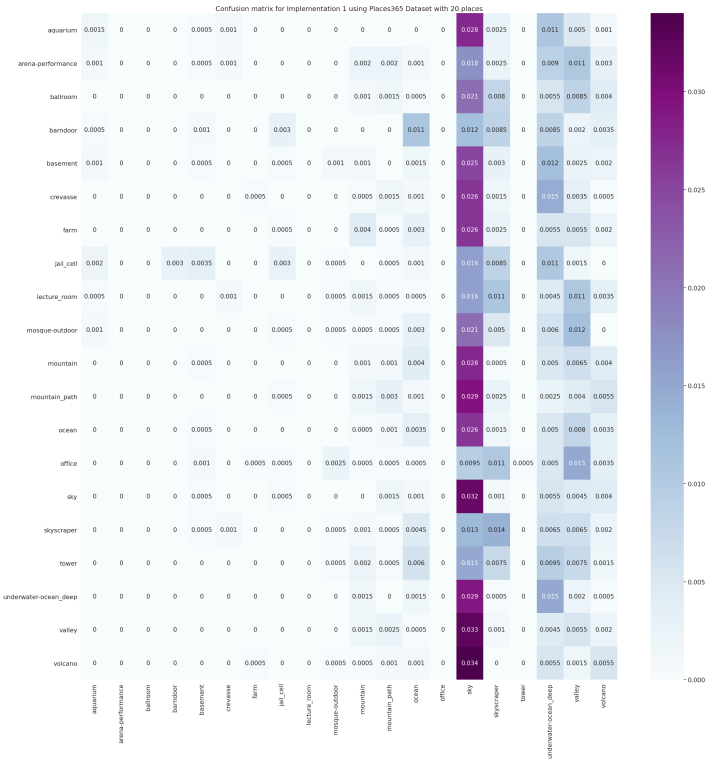


Figure 48: Topic Model Implementation 1 Prediction Confusion Matrix using new data on Places20 dataset

Classification Report : The Classification Report for the Baseline Model performance can be viewed in Figure 33.

7.4 Assessing the distribution of Topics in the Topic Model

After making predictions on the proposed model architectures, the internal working on the models are assessed for the same. What this basically means is checking the objects that are rendered while training the model and seeing whether the model effectively distributes the different objects highlighted by the softmax layer from the Resnet-50 when classifying the different batches of images into their respective classes.

One thing to note in this process is to see whether each Topic Dense layers categorizes the 1000 different objects differently. Each Topic dense layer is connected to the softmax layer of the backbone network such that each topic layer does not interact with each other. The EgoPlaces dataset is used for checking this experiment.

Given a number of dense layers where each layer represents 1000 objects, is

Places	precision	recall	f1-score	support
skyscraper	0.15	0.28	0.20	100.00
underwater-ocean_deep	0.10	0.30	0.15	100.00
sky	0.07	0.63	0.12	100.00
volcano	0.11	0.11	0.11	100.00
jail_cell	0.33	0.06	0.10	100.00
mountain_path	0.16	0.06	0.09	100.00
ocean	0.08	0.07	0.07	100.00
valley	0.04	0.11	0.06	100.00
aquarium	0.20	0.03	0.05	100.00
mountain	0.05	0.02	0.03	100.00
mosque-outdoor	0.08	0.01	0.02	100.00
basement	0.06	0.01	0.02	100.00
arena-performance	0.00	0.00	0.00	100.00
ballroom	0.00	0.00	0.00	100.00
barndoor	0.00	0.00	0.00	100.00
crevasse	0.00	0.00	0.00	100.00
farm	0.00	0.00	0.00	100.00
lecture_room	0.00	0.00	0.00	100.00
office	0.00	0.00	0.00	100.00
tower	0.00	0.00	0.00	100.00
accuracy				0.08
macro avg	0.07	0.08	0.05	2,000.00
weighted avg	0.07	0.08	0.05	2,000.00

Table 32: Classification Report for the Baseline Model using Places20 dataset

each topic layer of the model able to obtain different objects or whether they obtain the same 1000 objects with the same weights in each layer.

We assess this first by observing the top 10 objects identified by the softmax layer from Resnet-50(backbone) as well as the top 10 objects identified by each topic layer.

We then estimate the sum of the top 10 objects weight values for each topic and check the spread of the sum values for each topic layer. This is done by selecting 10 images from different classes checking the sum of the topics for each place label. Here we select the places namely:

- Office
- Pathways
- Transportation
- Restaurant, Bar
- Education, Science

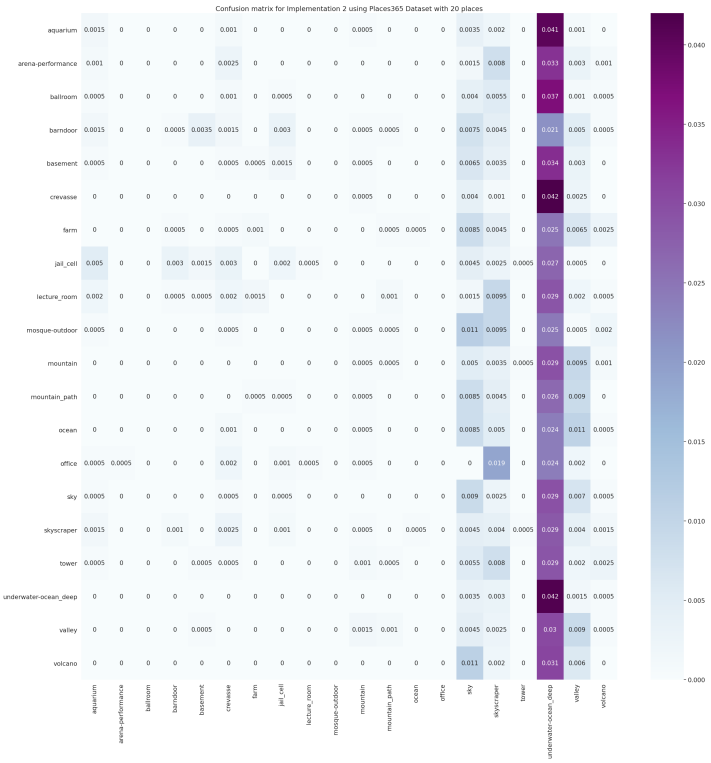


Figure 49: Topic Model Implementation 2 Prediction Confusion Matrix using new data on Places20 dataset

These place labels are selected as they are the places most commonly found in the EgoPlaces dataset. The methodology followed in order to estimate the sum of topic values is summarized in Figure 50. The values $T1$, $T2$, $T3$ and $T4$ in Figure 50 gives us the sum of the topics which is given by the equation :

$$T_i = \sum_{j=0}^{o} w_{j,i} * x_{j,i} \tag{18}$$

where j is the number of objects ranging from 0 to 999(this is from the 1000 objects from Resnet-50 softmax) and i is the topic number. The experiments are performed for four topics thus the maximum value of i is 4. Value w is the weight of the object and x is the value from the Softmax.

The methodology for assessing the topic distribution in each topic layer can be summarized as the following:

Sum of Top 10 Topic Layer values (Methodology 1)

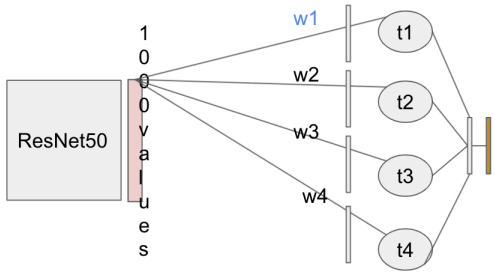
1. Select 5 classes

Places	Precision	Recall	F1-score	Support
valley	0.11	0.18	0.13	100.00
underwater-ocean_deep	0.07	0.83	0.13	100.00
sky	0.08	0.18	0.11	100.00
jail_cell	0.20	0.04	0.07	100.00
skyscraper	0.04	0.08	0.05	100.00
aquarium	0.10	0.03	0.05	100.00
farm	0.29	0.02	0.04	100.00
barndoor	0.09	0.01	0.02	100.00
mountain	0.07	0.01	0.02	100.00
arena-performance	0.00	0.00	0.00	100.00
ballroom	0.00	0.00	0.00	100.00
basement	0.00	0.00	0.00	100.00
crevasse	0.00	0.00	0.00	100.00
lecture_room	0.00	0.00	0.00	100.00
mosque-outdoor	0.00	0.00	0.00	100.00
mountain_path	0.00	0.00	0.00	100.00
ocean	0.00	0.00	0.00	100.00
office	0.00	0.00	0.00	100.00
tower	0.00	0.00	0.00	100.00
volcano	0.00	0.00	0.00	100.00
accuracy				0.07
macro avg	0.05	0.07	0.03	2,000.00
weighted avg	0.05	0.07	0.03	2,000.00

Table 33: Classification Report for Topic Model Implementation 2 using the Places20 dataset.

2. For each class, select 5 images at random.
3. For each image,
    - Estimate the values of $T1$,$T2$,$T3$ and $T4$ for only the top 10 objects for each topic(o=10).
    - Plot the sum for each topic as a box plot using these top 10 objects.

The above method gives the topic distribution per class. The box plots for the place labels listed previously for Topic Model Implementation 2 can be found in Figure 51. The models were also computed for Implementation 1. The sum of the 1000 objects for the softmax layer as well as the different topic layers can be found in Table 51 in the Appendix section of the report. Similarly, one can perform a similar experiment using the Top 10 objects from the Resnet-50 softmax and checking the distribution spread of the sum of topics. The methodology for this can be summarized below:



T1 = sum (w1,1*SOb1; w1,1000*SOb1000)

T2 = sum (w2,1*SOb1; w2,1000*SOb1000)

T3 = sum (w3,1*SOb1; w3,1000*SOb1000)

T4 = sum (w4,1*SOb1; w4,1000*SOb1000)

Figure 50: Formula for estimating the sum of the topic weights

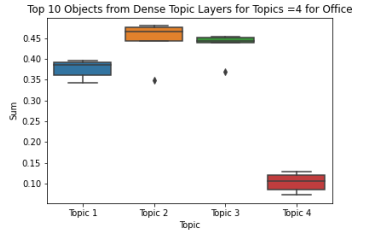
Sum of Top 10 Resnet-50 values (Methodology 2)

1. Select 5 classes
2. For each class, select 10 images.
3. For each image,
   - Estimate the top 10 objects of the Resnet-50 softmax layer.
   - Estimate the values of sum of topics $T1$, $T2$, $T3$ and $T4$ by only taking the product of the object weights(w) and the objects (o) which correspond to the Resnet-50 object values obtained in the previous step.
   - Plot the sum for each topic as a box plot using these top 10 objects.

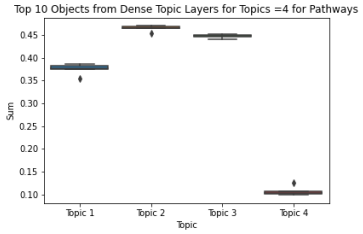
The box-plots obtained using this method for the 5 chosen places from the EgoPlaces dataset using Topic Model Implementation 2 can be found in Figure 52.

Worked out example for estimating the top 10 topics for Methodology 1 and 2 :

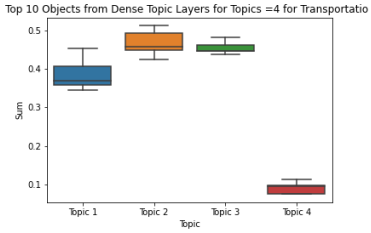
The different methodologies explained above can be further explained with the help of an example: For the example, we consider the sum of the top 4 objects from Resnet softmax as well as each of the topic layers for a single image. Say we have two topics $t1$ and $t2$. Let the top 5 objects highlighted for these 3 layers(two topic dense layers and Resnet-50 softmax layer) be the following:



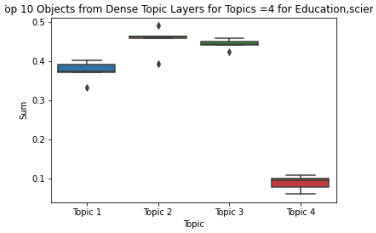
(a) Place : Office



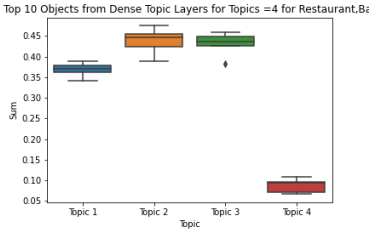
(b) Place : Pathways



(c) Place : Transportation



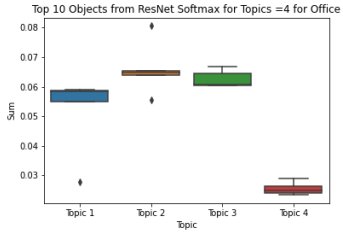
(d) Place : Education,Science



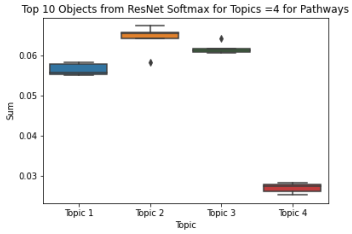
(e) Place : Restaurant,Bar

Figure 51: Sum of Top 10 Objects for each Topic Dense Layer (Methodology 1) for 5 place labels by taking 5 images belonging to a particular class(Implementation 2)

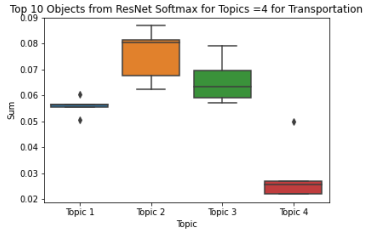
- **Top 4 Resnet-50 Objects** : 4,699,750 and 255
- **Resnet-50 object weights** : 0.25,0.1,0.045 and 0.005 respectively
- **Resnet-50 Softmax object value** : 0.15,0.25,0.05 and 0.75 respectively.
- **Top 4 Topic 1 Objects** : 699,555,224 and 229
- **Topic 1 object weights** : 0.3,0.12,0.005 and 0.0008.
- **Topic 1 object values** : 0.3,0.4,0.5,0.6 respectively
- **Top 4 Topic 2 Objects** : 4,699,750 and 420
- **Topic 2 object weights** : 0.45,0.155,0.01 and 0.007.



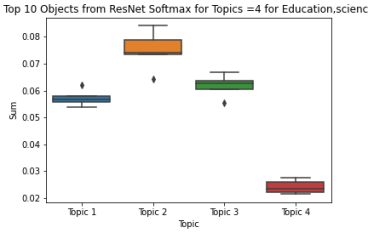
(a) Place : Office



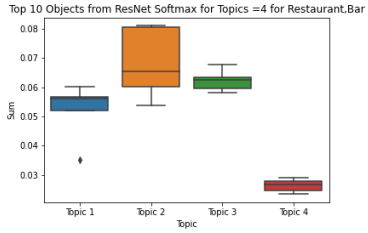
(b) Place : Pathways



(c) Place : Transportation



(d) Place : Education,Science



(e) Place : Restaurant,Bar

Figure 52: Sum of each Topic Dense Layer for top 10 Resnet-50 objects(Methodology 2) for 5 place labels by taking 10 images belonging to a particular class(Implementation 2)

- **Topic 2 object values** : 0.4,0.08,0.059,0.001 respectively.

With these values in mind, now we estimate the sum of the top 4 topics $t1$ and $t2$ for the topic layers(Methodology 1):

```
t1 = 0.3*0.3+0.12*0.4+0.005*0.5+0.0008*0.6 = 0.14
t2 = 0.45*0.4+0.155*0.08+0.01*0.059+0.007*0.001 = 0.192
```

Now we estimate the sum of the topic layers $t1$ and $t2$ for the top 4 Resnet values. For the sake of simplification, let us assume the values for the objects in the topic layer not present in the top 4 Resnet-50 are too minimal and are approximately zero. With this in mind, the sum of the topics for top 4 Resnet-50(Methodology 2) will give the following:

```
t1 = 0+0.3*0.3+0+0 = 0.09
```

```
t2 = 0.45*0.4+0.155*0.08+0.01*0.059+0.007*0 = 0.192
```

The values for the objects not present in the top 4 Resnet-50 softmax are considered to be zero. Thus this helps to view the distribution of the top 4 objects from the softmax layers in the respective topic dense layers. This method is repeated for each image of each place label(class). The values are then plotted to create the box plots as illustrated in Figures 51 and 52.

The above methodologies give the sum of top 10 topics for a number of images per class label. The sum of all the 1000 objects multiplied by a weight are given in Table 34. In this table, each column represents the sum of the product of the object values and their corresponding weights for the different topic layers as well as the Resnet-50 softmax layer.

For each place label, 5 images for a particular place label are considered. For each place, the average sum of topics for the topic layers(Topic 1, Topic 2, Topic 3, Topic 4 as well as the Resnet-50 softmax) are considered. It is observed that the sum of the 1000 objects from the Resnet-50 softmax comes out to be approximately equal to 1 whereas there is some variation in the sum of the product of the 1000 objects of the topic layers and their respective weight values. The Topic Model Implementation 2 was considered for the place labels : Office, Transportation, Education/science, Restaurant/Bar and Pathways from the EgoPlaces dataset. The images were sampled at random for each place label. The sum of the 1000 objects for Implementation 1 using the two methodologies discussed for the different topics layers as well as the softmax layer for the classes $underwater - ocean\_deep$, *sky* and *volcano* from the Places20 dataset can be found in Table 50 in the Appendix section of the report.

Place	**Renset**	**Topic 1**	**Topic 2**	**Topic 3**	**Topic 4**
Office	1	4.96	4.84	5.11	5.32
Transportation	1	5.411	5.212	5.59	5.73
Restaurant,Bar	1	4.877	4.754	5.03	5.29
Pathways	1	5.165	5.06	5.31	5.62
Education Science	1	5.017	4.91	5.23	5.47

Table 34: The average of the sum of 1000 topics for 5 images per place label using Topic Model Implementation 2 on EgoPlaces dataset

7.5 Plotting the different topics for the Topic Model

After analysing the functioning of the Topic Model, the next step would be to analyse the different objects rendered from the softmax layer from the Resnet-50 backbone network. As we consider all the 1000 objects as the output from a Resnet-50 network pre-trained using the ImageNet dataset, one can represent the different objects as the 1000 classes from the ImageNet dataset.

In order to check whether the model is functioning properly, the first step would be to view the different objects that are obtained from the softmax layer.

This can be performed by checking the different Topic dense layers as well as the Resnet-50 softmax layer itself. As all the different Topic Layers are connected to the output of the backbone softmax layer, the first test would be to see whether the different Topic layers perform differently from each other. This would be the expected behaviour as the layers are directly connected to the backbone network through the Resnet-50 output and contain no immediate connection to the other Topic dense layers.

This can be checked by loading a substantial amount of images belonging to a particular place label and seeing whether the objects obtained in the different Topic layers are different to each other or are similar. This can be performed by fitting the model with at most 1500 images for a particular place label and observing the results. This process can be repeated for all the Places from the dataset.

In order to perform the experiment, we consider the Topic Model Implementation 2 architecture and fit it with the required images and check the outputs of the Resnet-50 softmax layer along with the different Topic Dense layers.

After fitting the different images for one particular place label at a time, the different topics can be plotted with respect to the number of images. This is performed for all the Topic layers as well as the Softmax layer from the backbone network. One can then derive the top 10 objects with the highest image count for a particular place. This can be performed for different places. For our experiment, we will consider the places which have the highest number of images in the EgoPlaces dataset namely : **Office, Pathways and Transportation**. Due to size constraints the number of images per place label is capped to 1500 but this can be extended.

The top 10 frequent objects for 1500 for Office and Pathways can be viewed from Figures 53, 54, 55, 56 for Place : Office and Figures 57, 58, 59, 60 for Place : Pathways respectively. These objects occur the most among 1500 randomly sampled image for a particular place.

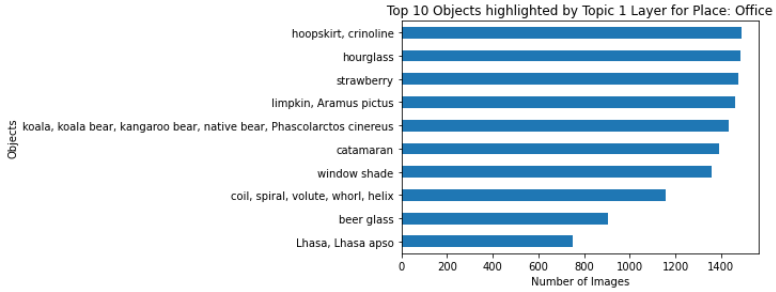


Figure 53: Top 10 occurring objects in Topic 1 layer for Place : Office

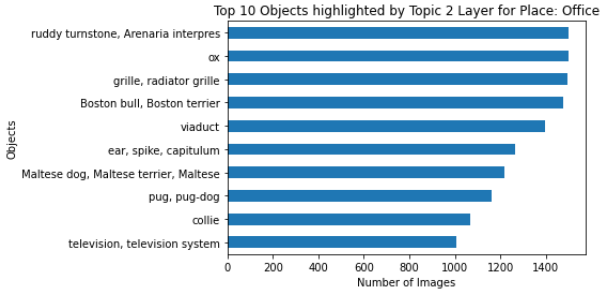


Figure 54: Top 10 occurring objects in Topic 2 layer for Place : Office

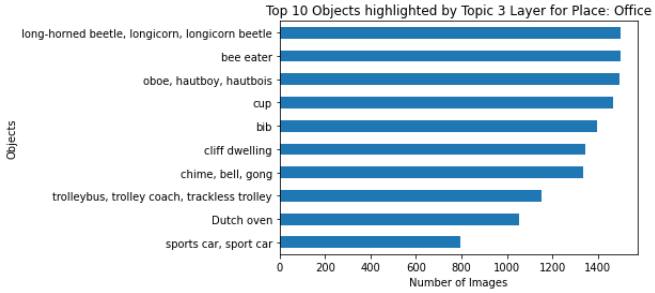


Figure 55: Top 10 occurring objects in Topic 3 layer for Place : Office

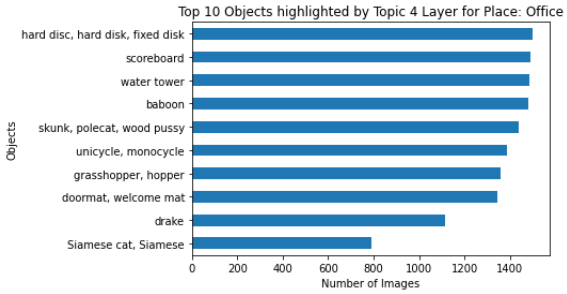


Figure 56: Top 10 occurring objects in Topic 4 layer for Place : Office

8 Conclusion

In this project, different model architectures were created in order to perform Place Classification using Topic Modelling. Some of these architectures were finalized and experiments were performed on these in order to compare the model performance with a baseline place classification model. The architectures as well as the results for the other models not selected can be found in Sections 11.1 and 11.2 of the Appendix.

The three selected model architectures namely the Baseline Model, Topic

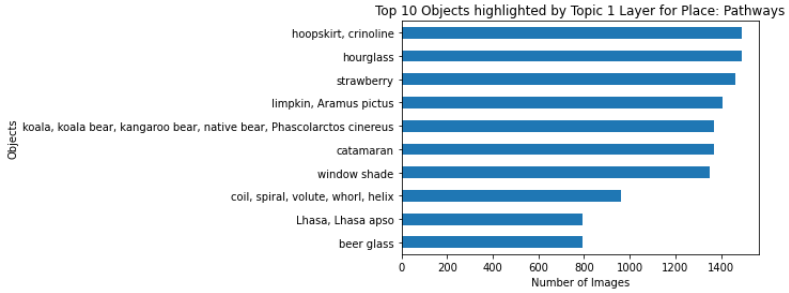


Figure 57: Top 10 occurring objects in Topic 1 layer for Place : Pathways

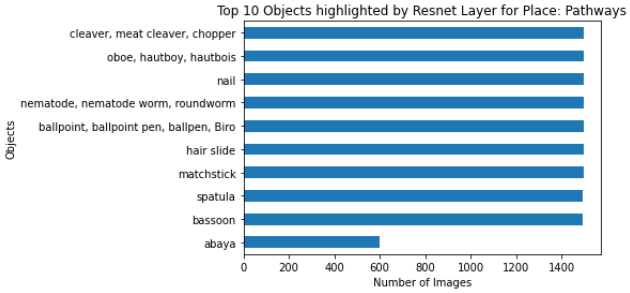


Figure 58: Top 10 occurring objects in Topic 2 layer for Place : Pathways

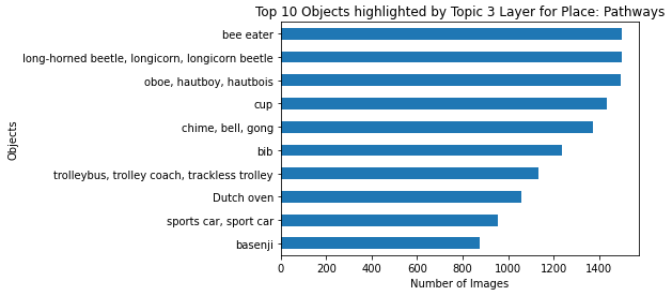


Figure 59: Top 10 occurring objects in Topic 3 layer for Place : Pathways

Model Implementation 1 and Topic Model Implementation 2 were trained using three datasets. These datasets were EgoPlaces, Places365 and Places20 which is a subset of the Places365 dataset with 20 place categories. These models were then used for performing place classification and the results were obtained using the datasets. Table 35 summarizes the three different architectures considered and their top-1 prediction accuracy for the different datasets.

From Table 35, it is quite evident that for the different datasets, the Baseline Place Classification model outperforms the topic models by a great extent.

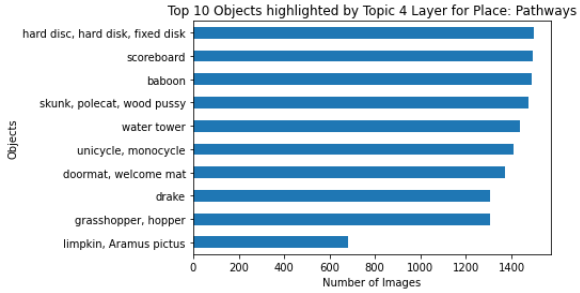


Figure 60: Top 10 occurring objects in Topic 4 layer for Place : Pathways

Dataset	Baseline	Topic Model 1	Topic Model 2
EgoPlaces with 24 places	42.6%	4%	4%
EgoPlaces without Place:Noise	28%	5.2%	12%
Places365 with 365 places	37.4%	0.5%	0.3%
Places20 with 20 places	75.2%	8%	7%

Table 35: Top-1 Model Prediction Accuracy for the model architecture using the different datasets

Additional experiments were performed in order to assess the effect of Noise on the predictions made by the different models. The performance of the baseline and the topic models can be found in detail in Appendix 11.3. This was done by eliminating the data elements in EgoPlaces with place labels sets as Noise. It is observed that in the case of Noise in the dataset, the majority of the images are classified as belonging to place Noise when using both the implementations of the Topic models. When the noise is eliminated, the mis-classifications made by the topic models are greatly reduced. This effect is less prominent in the case of the baseline models using the no noise dataset. Thus one can make the conclusion that the **Topic Models are greatly susceptible to Noise**.

Among the different Topic model implementations, it is observed that the **Implementation 1 Topic Model performs better than the Implementation 2 in most of the datasets** with the exception of the EgoPlaces dataset when Noise is removed.

Once the models are predicted upon, experiments were performed to evaluate the sum of the topics in the different Topic dense layers for all topics. This was performed by considering 5 images belonging to a particular place. The sum of the topics were obtained for the 1000 objects as well as the top 10 objects. The top 10 objects could be obtained as a result of the top 10 Reset-50 softmax objects(Methodology 2) with the highest values for a particular place label as well as the top-10 objects for each topic layer respectively(Methodology 1). It is observed that some topics have higher spread of the values for certain place categories than the others as can be viewed in Figures 51, 52 in the case of number of topics, k as 4 and Figures 72 and 73 when $k = 9$. This was performed

for both the EgoPlaces as well as the Places20 dataset. For each place for a particular dataset, the sum of the top 1000 values was close to 1 in the case of Resnet-50 softmax layer. While this was not the case in the respective Topic Dense layers(Refer Table 51). Figure 53 to 60 show the different topics which are obtained by mapping the ImageNet object labels to the objects of the Topic dense layers. Thus, this models does split the different objects obtained from the Resnet-50 layer into different topics such that these objects are quite different from one another. In other words, **the Topic Model does show a hint of Topic Modelling being performed**. Though it must be noted that the labels for the object values are crucial for Topic modelling in this case.

The Topic Model Implementation 1 was extended to the case when the number of topics, k was 9. This was done by using 9 dense layers for each topic rather than the previous 4 dense layers. The model performance with the 4 topic equivalent is summarized in Table 36.

Dataset	**k=4**	**k=9**
EgoPlaces without Place:Noise with 23 places	9%	14%
Places20 with 20 places	8%	2%

Table 36: Top-1 Prediction Accuracy for Topic Model Implementation 1 with 4 and 9 topics respectively.

The experiments for this can be found in Appendix 11.4 when the model is trained with the EgoPlaces dataset without Noise as well as the Places20 dataset. It is observed that **as the number of topics are increased, so do the number of trainable parameters.** However, the top-1 prediction accuracy of the model decreases as compared to the case of models with 4 topics using the Places20 dataset while the $k = 9$ Implementation of the Topic model performs better than the $k = 4$ equivalent. The box plots for the k=9 topics using Place : Transportation of the EgoPlaces dataset can be found in Figure 72 and the plots for Place : Sky of the Places20 dataset can be found in Figure 73. The different topics are summarized in Figures 74, 75,76, 77, 78,79, 80, 81, 82 for Place : sky of the Places-20 dataset in the Appendix (Section11.8).

The final observation about the experiments pertains to the places predicted by the model. From Figures 68, 48, 49 and 69, it can be observed that the **Place categories which are outdoor generally dominate the model prediction**. For example, the model tends to correctly classify a place such as Sky, Ocean-Underwater_Deep (Places20) and Transportation, Pathways (EgoPlaces) with much more confidence as compared to indoor places.

Thus in summary, the Topic Models do perform Topic modelling to an extent. However, **the current Topic model implementations are not well equipped enough to perform place classification for indoor as well as outdoor classes with high accuracy as compared to the base place classification model**. This means that there needs to be an additional learning function in the Topic Model layers to identify places more effectively. More about the suggested enhancements will be explained in the next section of the

report.

9 Suggested Enhancements and Future Work

This section deals with some suggested enhancements to the premise of the existing model architectures along with the scope of Future work that can be performed on Topic Modelling using this approach.

Extensive use of Data Augmentation techniques (referred in Section 5.2.1) may help in reducing the number of mis-classification in the model as this would prevent the model from over-fitting. This would involve using flipping and rotating the images at random as well as using data balancing techniques by means of data sampling such that the images from a particular place in the different datasets are represented equally. This is more important for EgoPlaces dataset as it is fairly unbalanced while Places365 and Places20 are balanced datasets.

Semantic Segmentation of the images can be leveraged in order to identify parts of the image as a bag of words(objects) in order to make the model learn better.The objects in an image can be represented as segments in the data. This approach is used frequently in the case of data segmentation and therefore can be extended to Place and Scene Recognition. Approaches towards Place Classification using this is found in Sections 2.1 and 2.3. Additionally, a weighing matrix logic can be applied to the objects in order to reward the frequently occurring objects for a particular place with a higher weight value which might increase the accuracy of correctly classifying the place by the model significantly.

The Topic Models can be emboldened by utilizing Long Short Term Memory (LSTM) and Recursive Neural Network(RNN) can help make the model learning process better through means of label embedding for a place. This approach can help detect objects in a continuous fashion. This would also help bring in a variable of time when finding the objects of an image or image frames in videos. This approach can then be compared to their CNN equivalents and observations can be made for their performance.

Once the accuracy of the model is brought up to a acceptable level, the model architecture construction can be streamlined through the means of sub-classing of the model layers. This is offered by Keras library in many programming languages. This can facilitate easy model deployment where-in the topic layers can be constructed automatically based on the parameter value for the number of topics(k) to be created as well as converting the number of units required in the top place layer as a parameter.

The final enhancement that can be made is by upgrading the multi-class classification problem to a multi-label classification problem can help identify the topics better as some objects may occur frequently in a combination of place categories. Through using a weighted hierarchy of objects with this approach, one can theorize better place predictions made by the models. The places could be categorized into indoor and outdoor categories which would help in preventing the model from generalizing the objects in an image.

10 Acknowledgements

During the course of the project, I received a lot of support and assistance.

Firstly I would like to thank my first supervisor Estefania Talavera Martinez for her unparalleled support and guidance throughout the course of the project. You provided me proper guidance which helped the project to progress in the right track as well as taking meetings in a prompt fashion to address pressing concerns. This project taught me a lot about the state of the art being applied in the field of Deep Learning in which I had limited knowledge in. I would like to thank my second supervisor Michael Biehl for letting me work on this project as well as supporting me throughout the duration of the project as well as getting me intrigued about the entire domain of Neural Networks as well as Deep Learning through his valuable lectures which helped solidify my grasp over the concepts.

I would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high performance computing cluster. This cluster helped me perform the majority of the experiments for the project. The rest of the projects were performed on Google Colaboratory for which I am grateful as well.

I would like to extend my gratitude to the creators the the different datasets for allowing me to perform research using them and have cited their work rightfully.

I would finally like to thank my family and friends. Their support and constant motivation helped me getting through this dissertation.

References

[1] 3.1. cross-validation: evaluating estimator performance¶.

[2] ece 20875 python for data science,n-grams and basic natural language processing. *ECE 20875 Python for Data Science,N-grams and basic natural language processing.*

[3] Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6).

[4] Layer information https://cs231n.github.io/convolutional-networks.

[5] Deep learning spreads, Feb 2018.

[6] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[7] D. Berrar. *Cross-Validation.* 01 2018.

[8] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 05 2003.

[9] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993, 01 2013.

[10] S. Bozinovski. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica (Slovenia)*, 44, 2020.

[11] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[12] F. Chollet et al. Keras, 2015.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] A. Dertat. Applied deep learning - part 4: Convolutional neural networks, Nov 2017.

[15] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[16] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[18] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[19] S. Ioffe. Probabilistic linear discriminant analysis. In A. Leonardis, H. Bischof, and A. Pinz, editors, *Computer Vision – ECCV 2006*, pages 531–542, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017.

[21] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[22] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[23] R. Mishra. Stratified-k-fold in machine learning, Mar 2021.

[24] P. Munro. *Backpropagation*, pages 73–73. Springer US, Boston, MA, 2010.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[26] S. Rani and P. Kumar. A sentiment analysis system for social media using machine learning techniques: Social enablement. *Digital Scholarship in the Humanities*, 34(3):569–581, 10 2018.

[27] P. Refaeilzadeh, L. Tang, and H. Liu. *Cross-Validation*, pages 532–538. Springer US, Boston, MA, 2009.

[28] D. Ribeiro. A practical introduction to deep learning with caffe and python, May 2019.

[29] S. Saha. A comprehensive guide to convolutional neural networks-the eli5 way, Dec 2018.

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[31] P. Singh. Probabilistic linear discriminant analysis (plda) explained url:https://towardsdatascience.com/probabilistic-linear-discriminant-analysis-plda-explained-253b5effb96, 2021.

[32] R. A. Suite. Introducing transfer learning as your next engine to drive future innovations, Feb 2020.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[34] E. Talavera, C. Wuerich, N. Petkov, and P. Radeva. Topic modelling for routine discovery from egocentric photo-streams. *Pattern Recognition*, page 107330, 2020.

[35] M. Talo. Convolutional neural networks for multi-class histopathology image classification. *ArXiv*, abs/1903.10035, 2019.

[36] M. Waskom, O. Botvinnik, D. O'Kane, P. Hobson, S. Lukauskas, D. C. Gemperline, T. Augspurger, Y. Halchenko, J. B. Cole, J. Warmenhoven, J. de Ruiter, C. Pye, S. Hoyer, J. Vanderplas, S. Villalba, G. Kunter, E. Quintero, P. Bachant, M. Martin, K. Meyer, A. Miles, Y. Ram, T. Yarkoni, M. L. Williams, C. Evans, C. Fitzgerald, Brian, C. Fonnesbeck, A. Lee, and A. Qalieh. mwaskom/seaborn: v0.8.1 (september 2017), Sept. 2017.

[37] L. Yang, X. Yan, C.-a. Zhang, and L. Wen. An ensemble convolutional neural networks for bearing fault diagnosis using multi-sensor data. *Sensors*, 19:5300, 12 2019.

[38] W.-H. Yeo, Y.-J. Heo, Y.-J. Choi, and B.-G. Kim. Place classification algorithm based on semantic segmented objects. *Applied Sciences*, 10(24), 2020.

[39] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

11 Appendix

11.1 Other Model Architectures

11.1.1 Implementation 3(Weighed Topic Model)

This implementation of the Topic model can also be termed as *Weighted Topic Model*. This implementation of the model makes use of a custom layer which is responsible for creation of the topics. The output from the Resnet-50 softmax layer is connected to this custom layer. The pseudo-code for the Topic Layer is provided below:

```
class Wt_Topics(keras.layers.Layer):
    def __init__(self, units):
        super(Wt_Topics, self).__init__()
        self.units = units

  def build(self,input_shape):
    self.w1 = self.add_weight(name = 'weight_topic_1',
        shape = (1, self.units),trainable = True,
        initializer = 'random_normal',
        regularizer = tf.keras.regularizers.l1_l2())

    self.w2 = self.add_weight(name = 'weight_topic_2',
        shape = (1, self.units),
        trainable = True,initializer = 'random_normal',
        regularizer = tf.keras.regularizers.l1_l2())
```

```
    self.w3 = self.add_weight(name = 'weight_topic_3',
    shape = (1, self.units),
    trainable = True,initializer = 'random_normal',
    regularizer = tf.keras.regularizers.l1_l2())

    self.w4 = self.add_weight(name = 'weight_topic_4',
    shape = (1, self.units),
    trainable = True,initializer = 'random_normal',
    regularizer = tf.keras.regularizers.l1_l2())

  def call(self, input1, input2, input3,input4):
    #Sum of the softmax output*topic weights
    return tf.multiply(input1,self.w1)+tf.multiply(input2, self.w2)+
        tf.multiply(input3, self.w3)+tf.multiply(input4, self.w4)
```

In the above defined class, there are four trainable weights which are responsible for evaluating the contribution of the respective topics when classifying a place for a batch of images. These weights are set to be trainable and are supplied by a regularizer term. The L1-L2 regularizer helps to penalize the weights in case of mis-classification. The output of this layer is given by the following equation:

$$topic\_layer = input * w_{t1} + input * w_{t2} + input * w_{t3} + input * w_{t4} \tag{19}$$

In the above equation, w_{t1}, w_{t2}, w_{t3}, w_{t4} are the weights of Topic 1, 2, 3 and 4 respectively and the input refers to the Softmax output from the ResNet network.

This architecture of the Topic Model is responsible for obtaining the contributions of the topic as a weighed sum. The architecture of this implementation of the topic model can be seen in Figure 61. The pseudo-code for the Topic

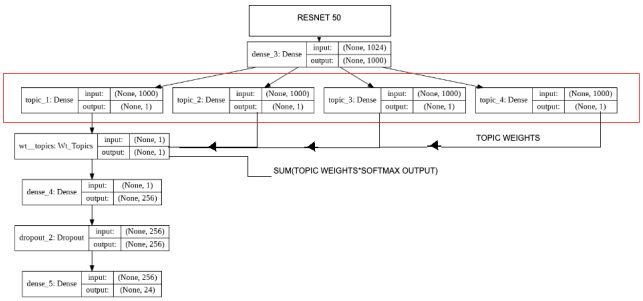


Figure 61: Topic Model Implementation 3 Architecture

Model implementation including calling the custom layer is given below:

```
    base_model = ResNet50(weights='imagenet', include_top=True,
```

```
                    input_shape=(224, 224, 3))
    wt_add = Wt_Topics(1000) #Calling the Topic Layer
    sum_layer = wt_add(base_model.output,base_model.output,
                    base_model.output,base_model.output)
    conc = keras.layers.Concatenate()(sum_layer)
    flat_2 = keras.layers.Flatten(name = 'flat_2')(conc)
    topic_flat = Dense(2000,name = 'first_dense')(flat_2)
    topic = Dropout(0.45,name = 'drop_1')(topic_flat)
    topic = Dense(256,activation='relu')(topic)
    topic = Dropout(0.2,name = 'drop_2')(topic)
    final = Dense(24,activation='softmax')(topic)

    cent_model = Model(inputs = base_model.input,outputs = final)
```

The implementation of the rest of this model implementation is similar to the Naive Topic Model implementation.

11.1.2 Implementation 4

Unlike the *Weighted Topic Model* defined previously, the main difference in this implementation is that the product of the Resnet model softmax layer and the individual topic weights are not summed together. The product is directly fed to the classification layers of the model. Thus this involved changing the Topic layer(custom layer) in order to perform the necessary task. The updated class definition can be found in the following pseudo-code :

```
class Wt_Topics(keras.layers.Layer):
    def __init__(self, units):
        super(Wt_Topics, self).__init__()
        self.units = units

  def build(self,input_shape):
    self.w1 = self.add_weight(name = 'weight_topic_1',
        shape = (1, self.units),trainable = True,
        initializer = 'random_normal',
        regularizer = tf.keras.regularizers.l1_l2())

    self.w2 = self.add_weight(name = 'weight_topic_2',
        shape = (1, self.units),
        trainable = True,initializer = 'random_normal',
        regularizer = tf.keras.regularizers.l1_l2())

    self.w3 = self.add_weight(name = 'weight_topic_3',
    shape = (1, self.units),
    trainable = True,initializer = 'random_normal',
    regularizer = tf.keras.regularizers.l1_l2())
```

```
    self.w4 = self.add_weight(name = 'weight_topic_4',
    shape = (1, self.units),
    trainable = True,initializer = 'random_normal',
    regularizer = tf.keras.regularizers.l1_l2())

  def call(self, input1, input2, input3,input4):
    return tf.multiply(input1,self.w1),tf.multiply(input2, self.w2),
        tf.multiply(input3, self.w3),tf.multiply(input4, self.w4)
```

The model architecture for this implementation of the Topic Model can be viewed from Figure 62. With the exception to the changes made to the Class

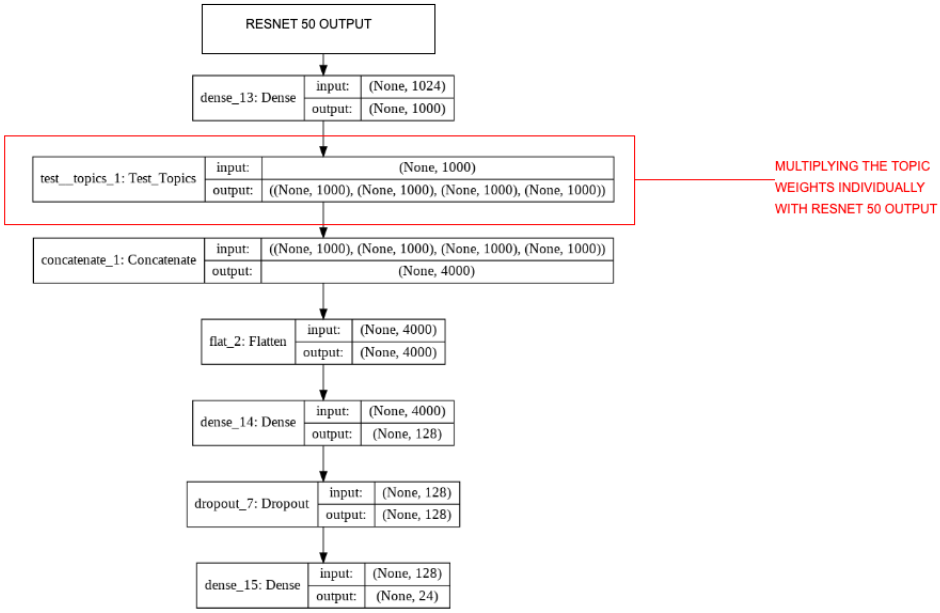


Figure 62: Topic Model Implementation 4 Architecture

definition of the Topic Class, the rest of the model architecture is similar to that of the Weighted Topic Model(Implementation 2) as shown by Figure 62.

11.1.3 Implementation 5

The fourth implementation of the model is based on the iteration of the Topic Model defined in 11.1.2.

This iteration is facilitated with the help of a Max Pooling layer with a stride of 4. The premise of this pooling layer is to obtain only 25% of the topics from the output of ResNet-50 backbone network. This in theory, would help to

significantly reduce the number of parameters in the network and help in faster predictions. The architecture of the network can be viewed from Figure 63.

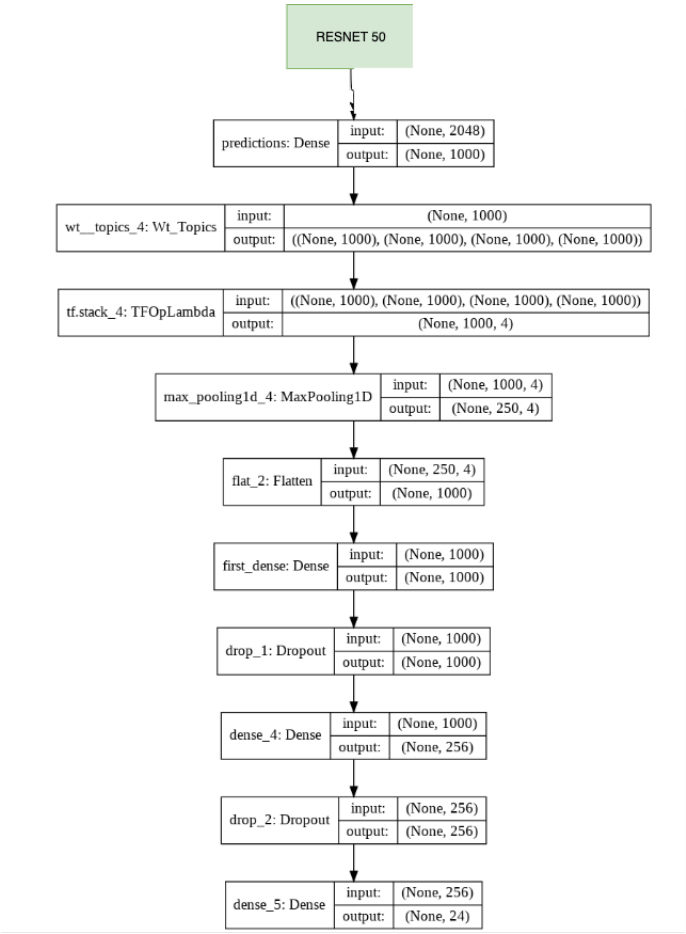


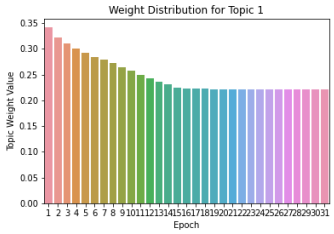
Figure 63: Topic Model Implementation 5 Architecture

Implementation 5 also makes use of lambda layers. This layer is responsible for pooling the 1000 values from ResNet to 250 values for each topic. Through the use of the *tf.stack()* function, the 4000 units(1000 per topic) are transformed to a shape of (4,1000). This layer then is fed to the Max Pooling layer which takes the 250 values from each topic, thus giving an output of shape (4,250) after pooling. This layer output is then flattened first, then fed to the classification dense layers similar to the way it is performed in prior implementations.

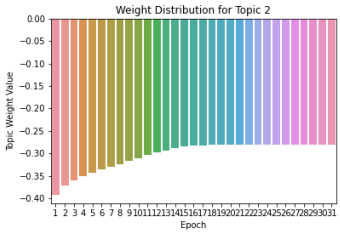
11.2 Assessing the performance of Weight Topic Model

11.2.1 Topic Weights without regularization

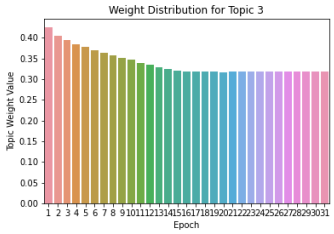
The topic model consists of 4 topic weights, one for each topic as discussed in the model architecture(Refer Figures 62,61,63). The weights of the model during model training without regularizing the weights is given by Figure 64. The weights remain the same after a certain epochs.



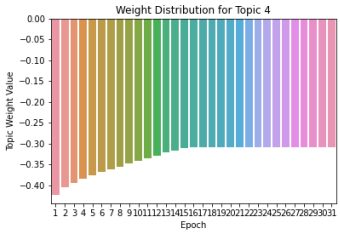
(a) Topic 1 weights



(b) Topic 2 weights



(c) Topic 3 weights



(d) Topic 4 weights

Figure 64: Topic weights for the Weighted Topic Model (Implementation 3 without regularizing weights)

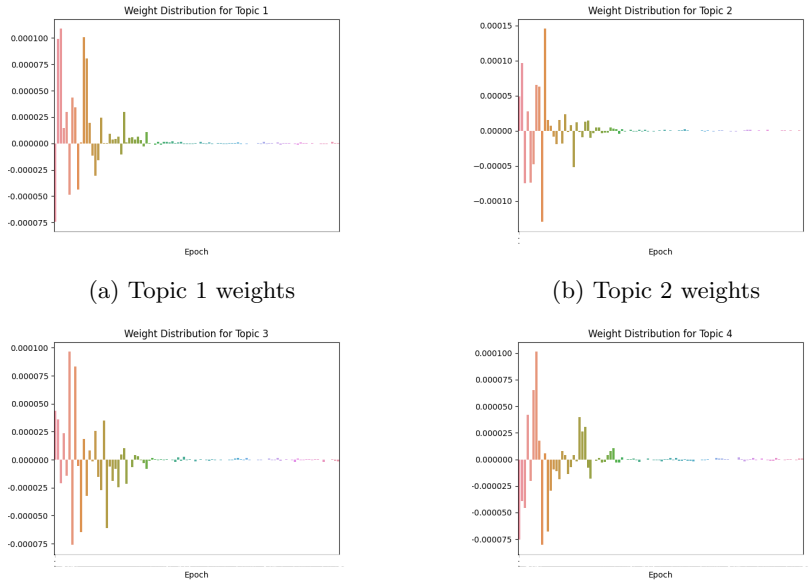
11.2.2 Topic Weights with regularization

Now the L1_L2() regularizers are used and the model is trained again. Figure 65 summarizes the topic weight distributions in this case.

We observe that the weights fluctuate after every epochs and some topics have counter weight distributions as compared to the other topic layers. However, the weights become too small after a certain number of epochs.

11.2.3 Problem with the model implementation with the weight layer

The Model Implementations 3,4 and 5 utilize the weight Topic Class which provides a Topic weight for each topic layer in the model. We now plot the accuracy and the model loss for the same and make observations. The loss and the accuracy for Topic Model Implementation 3 are given in Figure 66.



(a) Topic 1 weights

(b) Topic 2 weights

(c) Topic 3 weights

(d) Topic 4 weights

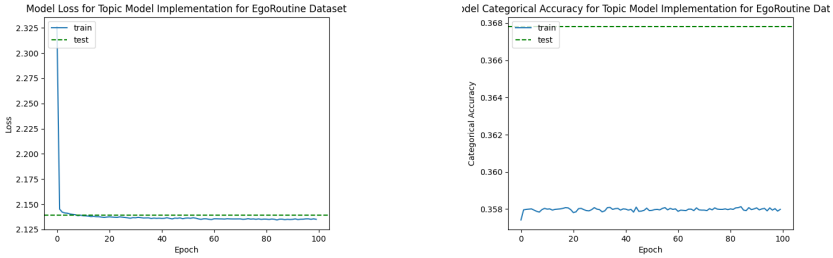
Figure 65: Topic weights for the Weighted Topic Model (Implementation 3 with f1_f2 regularizer on the Topic weights)

These models suffer from oscillating loss as can be viewed from the figure. The accuracy of the model is the highest at 36.80% and keeps oscillating between this value when trained. During prediction, the model is not able to identify any place correctly. This is due to the limitation of the manner how the weights are distributed in the model. Thus this weight class approach was discarded. This was the same case with Implementations 4 and 5 as well. The weight class was then replaced with fully connected layers for each topic. This is present in Model Implementation 1 and 2.

11.3 Assessing the effect of Place:Noise on the models

After performing model prediction on the EgoPlaces datasets for the Baseline Model as well as the different Topic Models, it is observed that in the case of Topic Models,the models are susceptible to Noise. The majority of predictions made by the Topic Models result in mis-classifying the images as Noise though that is less prevalent in the case of the Baseline Model.

Thus, we eliminate the effect of Noise by eliminating all the images with place category as Noise from the dataset. This dataset is then used to train and predict the Baseline and the Topic Models. The experiment parameters as well as the results during model prediction,training and testing are listed in this section



(a) Model Implementation 3 Loss (b) Implementation Model 3 Accuracy

Figure 66: Topic weights for the Weighted Topic Model (Implementation 3 with f1_f2 regularizer on the Topic weights)

for the different Topic Models. This dataset contains 23 place categories instead of the 24 places listed by the EgoPlaces dataset.

11.3.1 Baseline Model

The parameters and the metric values for the Baseline model during model training and evaluation are summarized in Tables 37 and 38 respectively. The Confusion Matrix for the baseline can be found in Figure 67.

Parameters	**Value**
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01
Training Images Count	54017
Validation Images Count	7029
Testing Images Count	16368
Number of epochs	40
Count of Places	23
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	53,120

Table 37: Parameters used for training Baseline Model for EgoPlaces dataset without Place : Noise.

Metrics	Train	Val	Test
Loss	0.0002	0.7571	0.7437
Categorical Accuracy	100%	91.64%	91.82%
Precision	100%	91.90%	92.2%
Recall	99%	91.48%	91.69%

Table 38: Parameters used for training Baseline Model for EgoPlaces dataset without Place : Noise.

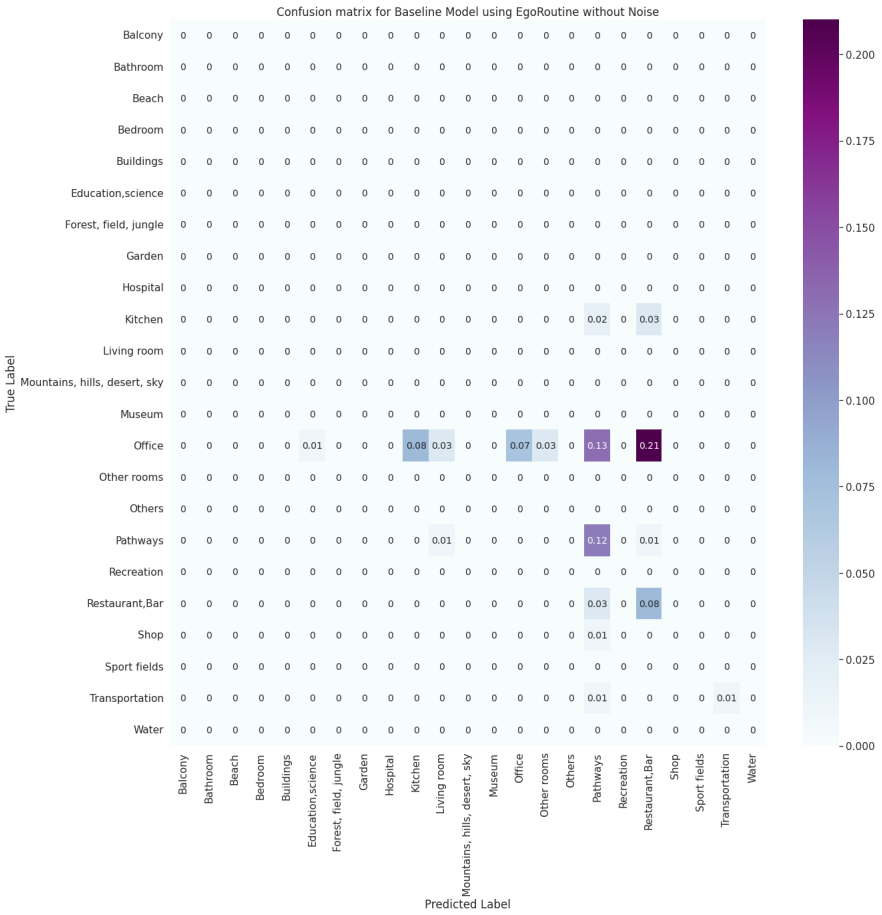


Figure 67: Baseline Model Prediction Confusion Matrix using new data on EgoPlaces dataset without Place : Noise

11.3.2 Topic Model Implementation 1

The parameters and the metric values for the Topic model implementation 1 during model training and evaluation are summarized in Tables 39 and 40 re-

spectively. The Confusion Matrix for this experiment can be found in Figure 68.

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01
Training Images Count	54017
Validation Images Count	7029
Testing Images Count	16368
Number of epochs	130
Count of Places	23
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	5,030,167

Table 39: Parameters used for training Topic Model Implementation 1 for Ego-Places dataset without Place : Noise.

Metrics	Train	Val	Test
Loss	0.6438	1.1090	1.1154
Categorical Accuracy	80.26%	69.72%	69.37%
Precision	90.78%	81.08%	80.83%
Recall	72.53%	61.95%	61.50%

Table 40: Parameters used for training Topic Model Implementation 1 for Ego-Places dataset without Place : Noise.

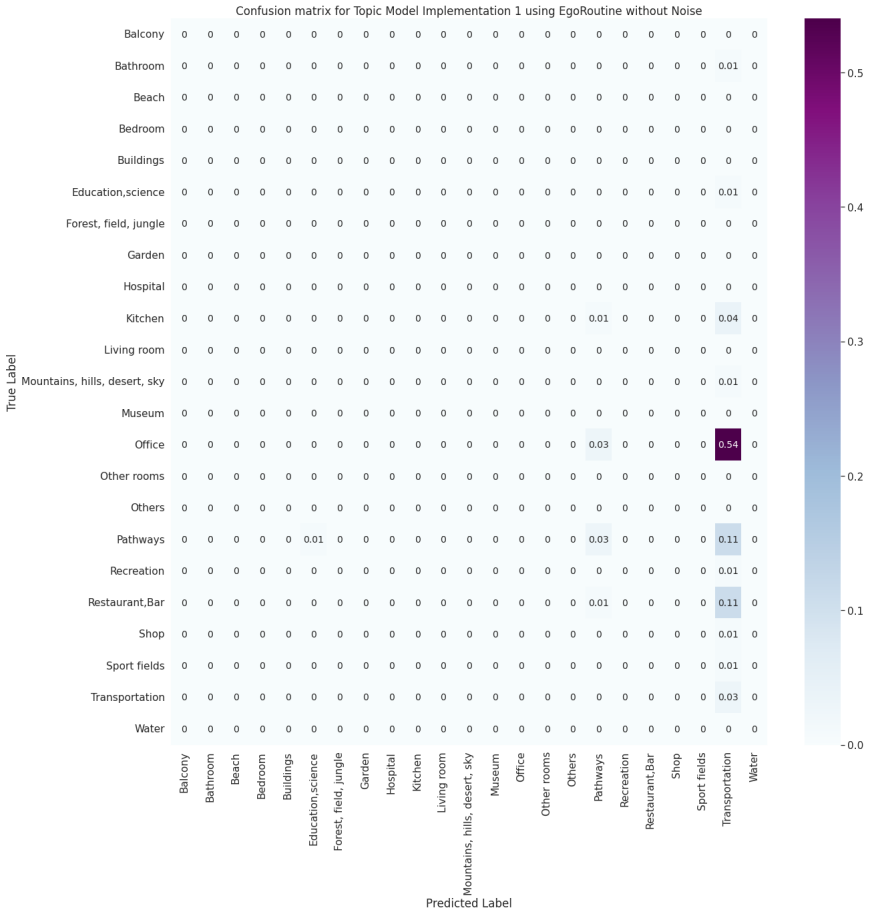


Figure 68: Topic Model Implementation 1 Prediction Confusion Matrix using new data on EgoPlaces dataset without Place : Noise

11.3.3 Topic Model Implementation 2

The parameters and the metric values for the Topic model implementation 2 during model training and evaluation are summarized in Tables 41 and 41 respectively. The Confusion Matrix for the Topic Model Implementation 2 can be found in Figure 69.

11.4 Creating Topic Models with 9 Topics

11.4.1 Introduction

In this section, we deal with Topic Models with 9 topics. This section deals with training the Topic Model Implementation 1 when the value of the Topics is 9 instead of 4 as discussed previously. The models are trained and evaluated for

Parameters	Value
Training Batch Size	151
Validation Batch Size	128
Testing Batch Size	128
Learning Rate	0.01
Training Images Count	54017
Validation Images Count	7029
Testing Images Count	16368
Number of epochs	40
Count of Places	23
Optimizer Used	Adam
Loss Metrics	Categorical Cross-Entropy
Validation Metrics	Categorical Accuracy,Precision,Recall
Number of Trainable Parameters	11,265,167

Table 41: Parameters used for training Topic Model Implementation 2 for EgoPlaces dataset without Place : Noise.

Metrics	Train	Val	Test
Loss	0.4167	1.6856	1.6947
Categorical Accuracy	85.62%	68.97%	68.37%
Precision	91.35%	75.25%	74.81%
Recall	81.43%	65.36%	64.60%

Table 42: Parameters used for training Topic Model Implementation 2 for EgoPlaces dataset without Place : Noise.

the EgoPlaces as well as the Places20 dataset. The performance of the model is then compared to the same implementation with 4 topics to see whether the topic modelling improves with the number of topics for different datasets. In order to utilize the first implementation of the Topic model for 9 topics, the model architecture is modified in order to solve this problem. This is done by using 9 dense layers serving as Topic layers instead of the 4 dense layers used previously. The remaining model remains the same.

11.4.2 Metric values during training and evaluation

In order to check the model performance in the case of nine topics, the same metrics namely : Categorical Accuracy,Precision and Recall are utilized. These are the same metrics used to assess the performance of the previous model architectures. Tables 43 and 44 summarize the results for the Topic Model with the metric values previously stated for both the EgoPlaces and the Places20 dataset respectively.

For the Places20 dataset, the model is trained for 100 epochs. The model is trained for 50 epochs in the case of EgoPlaces dataset. The model contains

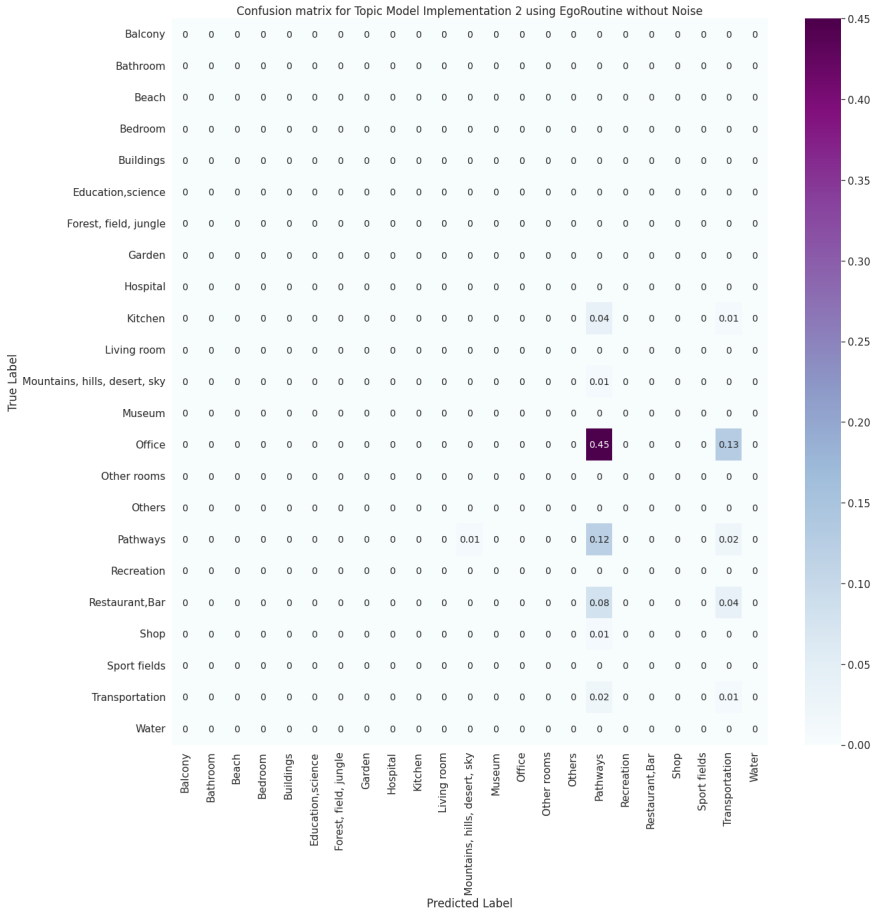


Figure 69: Topic Model Implementation 1 Prediction Confusion Matrix using new data on EgoPlaces dataset without Place : Noise

13, 618, 772 trainable parameters which is substantially higher to their four topic counterparts. The learning rate of the model is set to about 0.1.

Metrics	**Train**	**Val**	**Test**
Loss	0.9542	1.0896	1.1126
Categorical Accuracy	70.99%	68.45%	67.67%
Precision	85.81%	80.49%	80.04%
Recall	60.07%	59.79%	59.44%

Table 43: Parameters used for training Topic Model Implementation 1 with 9 topics for EgoPlaces dataset without Place : Noise.

Metrics	Train	Test
Loss	0.9507	2.4698
Categorical Accuracy	67.98%	38.65%
Precision	85.04%	50.62%
Recall	54.25%	28.45%

Table 44: Parameters used for training Topic Model Implementation 1 with 9 topics for Places20 dataset.

11.4.3 Assessing the Model Prediction Performance

After training the model over the two datasets, the model is then tested for prediction over the two datasets. Figures 70 and 71 respectively gives us the prediction performance for the Topic Model Implementation 1 on the EgoPlaces and the Places20 datasets. The accuracy of the model is about 2% which is significantly lower than the same model with 4 topics in the case of EgoPlaces dataset. For the Places20 dataset, the model exhibits around % top-1 prediction accuracy.

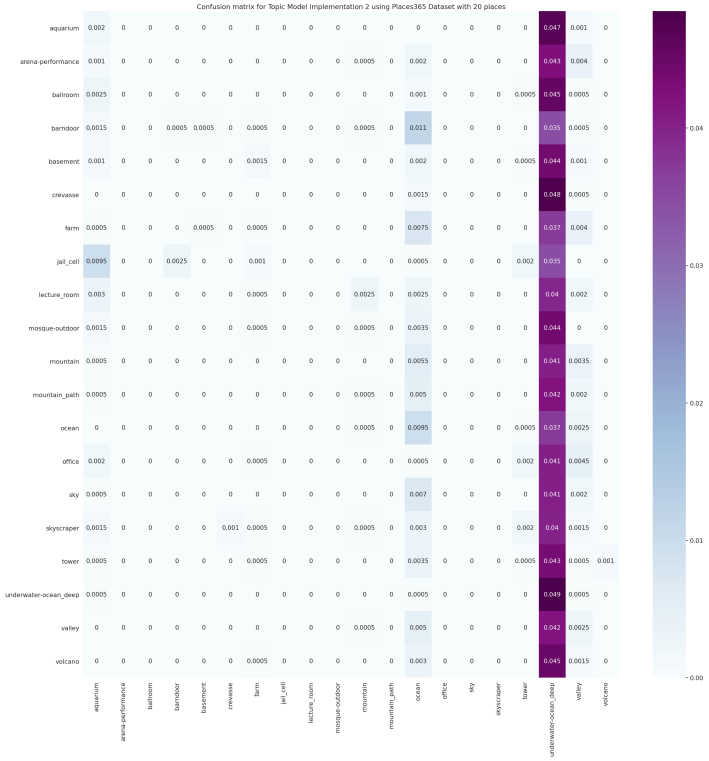


Figure 70: Topic Model Implementation 1 Prediction Confusion Matrix using non-train data on EgoPlaces dataset without Place : Noise

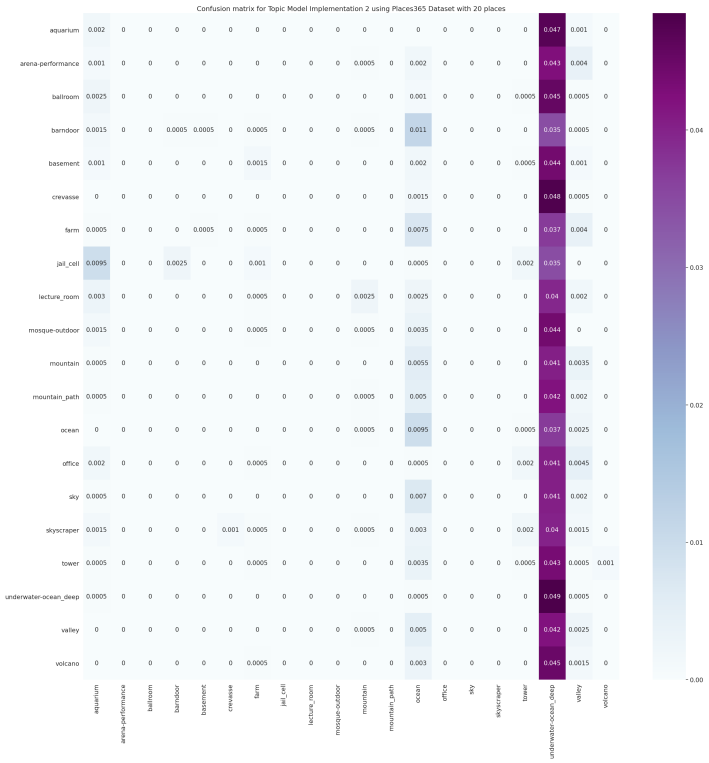


Figure 71: Topic Model Implementation 1 Prediction Confusion Matrix using testing data on Places20 dataset

11.4.4 Plotting the box plots for the Topic Model Implementation 1

Using the topic assessment methodology discussed in Section 7.4, we now plot the Topic Model Implementation 1 box plot for the top 10 objects identified by the Resnet softmax layer(Figures 72a,73a) according to Methodology 1 as well as the top 10 objects identified by each of the 9 topic layers(Figures 72b and 73b) according to Methodology 2.

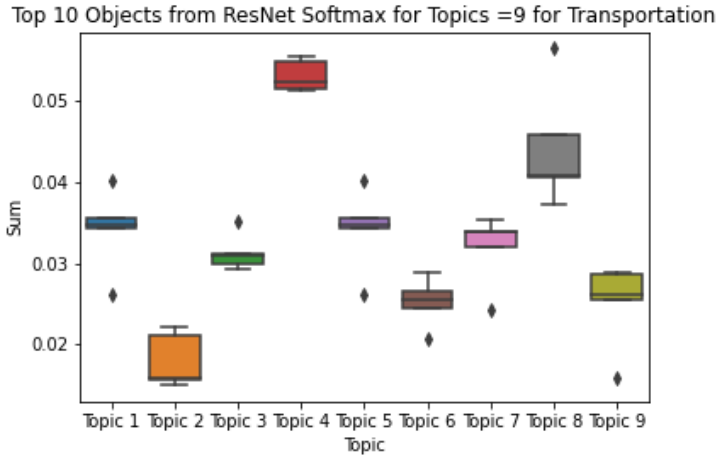
11.5 Classification Report Places365

11.5.1 Baseline Model

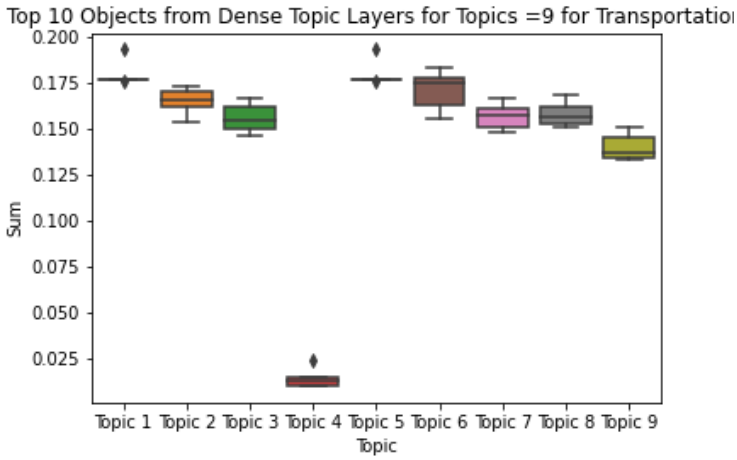
The classification report for the Baseline Model using the Places365 dataset can be found in Table 45. Here the places which exhibit an f-1 score of over 60% are displayed.

Place	precision	recall	f1-score	support
car_interior	0.752	0.880	0.811	100.000
ball_pit	0.857	0.720	0.783	100.000
cockpit	0.743	0.810	0.775	100.000
gymnasium-indoor	0.738	0.760	0.749	100.000
army_base	0.700	0.770	0.733	100.000
bus_station-indoor	0.823	0.650	0.726	100.000
boxing_ring	0.727	0.720	0.724	100.000
basketball_court-indoor	0.624	0.830	0.712	100.000
wind_farm	0.734	0.690	0.711	100.000
underwater-ocean_deep	0.750	0.660	0.702	100.000
carrousel	0.879	0.580	0.699	100.000
pizzeria	0.759	0.630	0.689	100.000
windmill	0.643	0.720	0.679	100.000
water_tower	0.667	0.660	0.663	100.000
phone_booth	0.713	0.620	0.663	100.000
bowling_alley	0.738	0.590	0.656	100.000
arena-hockey	0.595	0.720	0.652	100.000
escalator-indoor	0.563	0.760	0.647	100.000
volleyball_court-outdoor	0.674	0.620	0.646	100.000
bullring	0.629	0.660	0.644	100.000
racecourse	0.646	0.640	0.643	100.000
bus_interior	0.565	0.740	0.641	100.000
laundromat	0.747	0.560	0.640	100.000
landing_deck	0.627	0.640	0.634	100.000
wave	0.667	0.600	0.632	100.000
raceway	0.531	0.770	0.629	100.000
raft	0.535	0.760	0.628	100.000
bamboo_forest	0.550	0.710	0.620	100.000
crevasse	0.671	0.570	0.616	100.000
igloo	0.593	0.640	0.615	100.000
hangar-indoor	0.521	0.740	0.612	100.000
florist_shop-indoor	0.510	0.730	0.601	100.000

Table 45: Places365 Classification Report for Baseline Model with F1-Score greater than 0.6



(a) Top 10 Resnet objects Place : Transportation



(b) Top 10 objects from each Topic Layer Place : Transportation

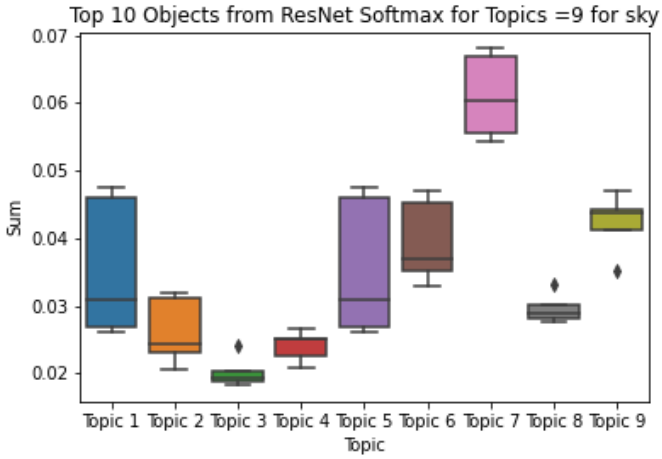
Figure 72: Box plots for sum of Top 10 Objects for each for Place : Transportation using Implementation 1 for 9 Topics with EgoPlaces dataset

11.5.2 Topic Model Implementation 2

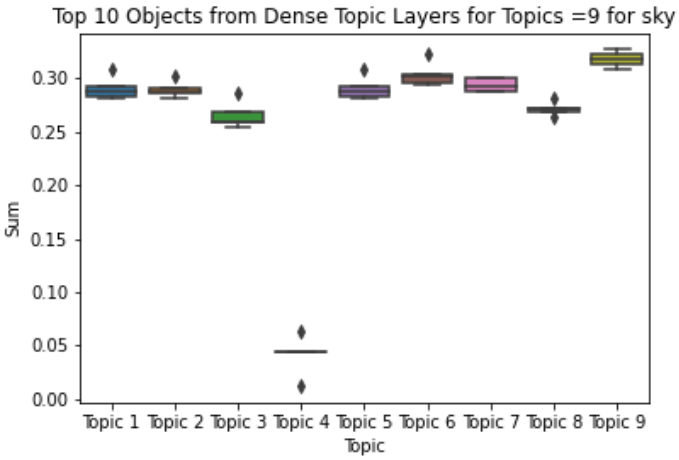
The classification report for the Topic Model Implementation 2 using the Places365 dataset can be found in Table 45.

Place	Precision	Recall	F1-score	Support
ocean	0.016	0.030	0.021	100.000
jail_cell	1.000	0.010	0.020	100.000
home_theater	0.500	0.010	0.020	100.000
barndoor	0.333	0.010	0.019	100.000
tower	0.143	0.010	0.019	100.000
arena-performance	0.125	0.010	0.019	100.000
volcano	0.009	0.320	0.018	100.000
crevasse	0.010	0.020	0.013	100.000
skyscraper	0.008	0.020	0.011	100.000
mountain_path	0.013	0.010	0.011	100.000
mosque-outdoor	0.006	0.020	0.009	100.000
underwater-ocean_deep	0.004	0.400	0.008	100.000
mountain	0.004	0.390	0.008	100.000
sky	0.006	0.010	0.007	100.000
valley	0.004	0.080	0.007	100.000
tree_farm	0.002	0.010	0.004	100.000
basement	0.002	0.010	0.003	100.000
aquarium	0.001	0.020	0.003	100.000
wind_farm	0.001	0.030	0.002	100.000
macro avg	0.006	0.004	0.001	36500.000
weighted avg	0.006	0.004	0.001	36500.000
accuracy	0.004	0.004	0.004	0.004

Table 46: Full Classification Report for Topic Model Implementation 2 using Places365 dataset



(a) Top 10 Resnet-50 objects.Place : Sky



(b) Top 10 topic layer objects.Place : Sky

Figure 73: Box plots for sum of Top 10 Objects for each for Place : Sky using Implementation 1 for 9 Topics with Places20 dataset

11.6 Class Labels for Places365 dataset (class 87-364)

The Class labels for the Places365 dataset can be found in Tables 47, 48 and 49.

PLACES		
87. chalet	88. chemistry_lab	89. childs_room
90. church/indoor	91. church/outdoor	92. classroom
93. clean_room	94. cliff	95. closet
96. clothing_store	97. coast	98. cockpit
99. coffee_shop	100. computer_room	101. conference_center
102. conference_room	103. construction_site	104. corn_field
105. corral	106. corridor	107. cottage
108. courthouse	109. courtyard	110. creek
111. crevasse	112. crosswalk	113. dam
114. delicatessen	115. department_store	116. desert/sand
117. desert/vegetation	118. desert_road	119. diner/outdoor
120. dining_hall	121. dining_room	122. discotheque
123. doorway/outdoor	124. dorm_room	125. downtown
126. dressing_room	127. driveway	128. drugstore
129. elevator/door	130. elevator_lobby	131. elevator_shaft
132. embassy	133. engine_room	134. entrance_hall
135. escalator/indoor	136. excavation	137. fabric_store
138. farm	139. fastfood_restaurant	140. field/cultivated
141. field/wild	142. field_road	143. fire_escape
144. fire_station	145. fishpond	146. flea_market/indoor
147. florist_shop/indoor	148. food_court	149. football_field
150. forest/broadleaf	151. forest_path	152. forest_road
153. formal_garden	154. fountain	155. galley
156. garage/indoor	157. garage/outdoor	158. gas_station
159. gazebo/exterior	160. general_store/indoor	161. general_store/outdoor
162. gift_shop	163. glacier	164. golf_course
165. greenhouse/indoor	166. greenhouse/outdoor	167. grotto
168. gymnasium/indoor	169. hangar/indoor	170. hangar/outdoor
171. harbor	172. hardware_store	173. hayfield

Table 47: Place Labels in Places365 Dataset (Part 2)

PLACES		
174. heliport	175. highway	176. home_office
177. home_theater	178. hospital	179. hospital_room
180. hot_spring	181. hotel/outdoor	182. hotel_room
183. house	184. hunting_lodge/outdoor	185. ice_cream_parlor
186. ice_floe	187. ice_shelf	188. ice_skating_rink/indoor
189. ice_skating_rink/outdoor	190. iceberg	191. igloo
192. industrial_area	193. inn/outdoor	194. islet
195. jacuzzi/indoor	196. jail_cell	197. japanese_garden
198. jewelry_shop	199. junkyard	200. kasbah
201. kennel/outdoor	202. kindergarden_classroom	203. kitchen
204. lagoon	205. lake/natural	206. landfill
207. landing_deck	208. laundromat	209. lawn
210. lecture_room	211. legislative_chamber	212. library/indoor
213. library/outdoor	214. lighthouse	215. living_room
216. loading_dock	217. lobby	218. lock_chamber
219. locker_room	220. mansion	221. manufactured_home
222. market/indoor	223. market/outdoor	224. marsh
225. martial_arts_gym	226. mausoleum	227. medina
228. mezzanine	229. moat/water	230. mosque/outdoor
231. motel	232. mountain	233. mountain_path
234. mountain_snowy	235. movie_theater/indoor	236. museum/indoor
237. museum/outdoor	238. music_studio	239. natural_history_museum
240. nursery	241. nursing_home	242. oast_house
243. ocean	244. office	245. office_building
246. office_cubicles	247. oilrig	248. operating_room
249. orchard	250. orchestra_pit	251. pagoda
252. palace	253. pantry	254. park
255. parking_garage/indoor	256. parking_garage/outdoor	257. parking_lot

Table 48: Place Labels in Places365 Dataset (Part 3)

PLACES		
258. pasture	259. patio	260. pavilion
261. pet_shop	262. pharmacy	263. phone_booth
264. physics_laboratory	265. picnic_area	266. pier
267. pizzeria	268. playground	269. playroom
270. plaza	271. pond	272. porch
273. promenade	274. pub/indoor	275. racecourse
276. raceway	277. raft	278. railroad_track
279. rainforest	280. reception	281. recreation_room
282. repair_shop	283. residential_neighborhood	284. restaurant
285. restaurant_kitchen	286. restaurant_patio	287. rice_paddy
288. river	289. rock_arch	290. roof_garden
291. rope_bridge	292. ruin	293. runway
294. sandbox	295. sauna	296. schoolhouse
297. science_museum	298. server_room	299. shed
300. shoe_shop	301. shopfront	302. shopping_mall/indoor
303. shower	304. ski_resort	305. ski_slope
306. sky	307. skyscraper	308. slum
309. snowfield	310. soccer_field	311. stable
312. stadium/baseball	313. stadium/football	314. stadium/soccer
315. stage/indoor	316. stage/outdoor	317. staircase
318. storage_room	319. street	320. subway_station/platform
321. supermarket	322. sushi_bar	323. swamp
324. swimming_hole	325. swimming_pool/indoor	326. swimming_pool/outdoor
327. synagogue/outdoor	328. television_room	329. television_studio
330. temple/asia	331. throne_room	332. ticket_booth
333. topiary_garden	334. tower	335. toyshop
336. train_interior	337. train_station/platform	338. tree_farm
339. tree_house	340. trench	341. tundra
342. underwater/ocean_deep	343. utility_room	344. valley
345. vegetable_garden	346. veterinarians_office	347. viaduct
348. village	349. vineyard	350. volcano
351. volleyball_court/outdoor	352. waiting_room	353. water_park
354. water_tower	355. waterfall	356. watering_hole
357. wave	358. wet_bar	359. wheat_field
360. wind_farm	361. windmill	362. yard
363. youth_hostel	364. zen_garden	

Table 49: Place Labels in Places365 Dataset (Part 4)

Resnet	Topic 1	Topic 2	Topic 3	Topic 4	Class
1	5.47	5.56	5.428	5.65	underwater-ocean_deep
1	5.503	5.713	5.55	5.908	sky
1	5.37	5.56	5.4407	5.73	volcano

Table 50: Sum of the 1000 objects for the Topic Layers and the Resnet-50 output layer for Implementation 1

Resnet	Topic 1	Topic 2	Topic 3	Topic 4	Class
1	3.918	4.223	3.916	3.745	Pathways
1	3.95	4.226	3.77	3.66	Education,Science
1	3.906	4.16	3.89	3.66	Office
1	3.93	4.221	3.929	3.715	Restaurant,Bar
1	3.702	3.974	3.749	3.514	Transportation

Table 51: Sum of the 1000 objects for the Topic Layers and the Resnet-50 outputlayer for Implementation 1 using EgoPlaces dataset

11.7 Other experiments

The Sum of the 1000 objects for the Topic Layers and the Resnet-50 output layer for Implementation 1 for Places20 dataset can be found in Table 50. Sum of the 1000 objects for the Topic Layers and the Resnet-50 output layer for Implementation 1 using EgoPlaces dataset can be found in Table 51.

11.8 Topics generated using Topic Model with 9 Topics

The Topics for place sky for the $k = 9$ Topic Model implementations are given in the Figures 74,75,76,77,78,79,80,81,82.

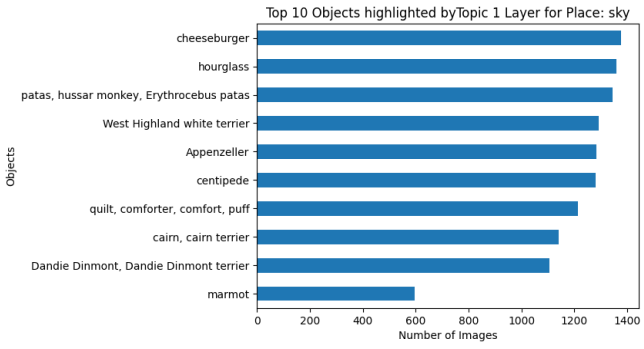


Figure 74: Top 10 occurring objects in Topic 1 layer

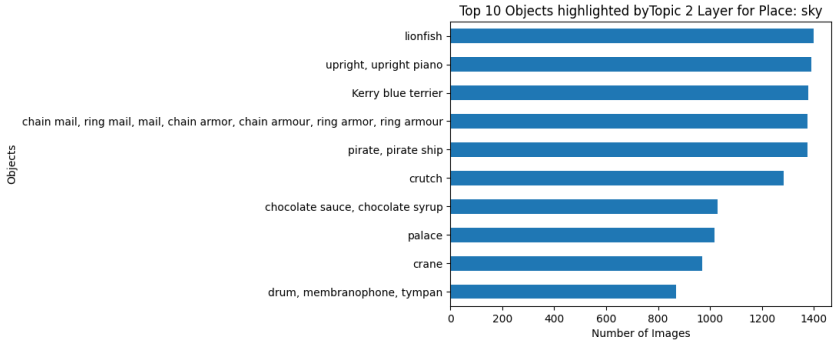


Figure 75: Top 10 occurring objects in Topic 2 layer

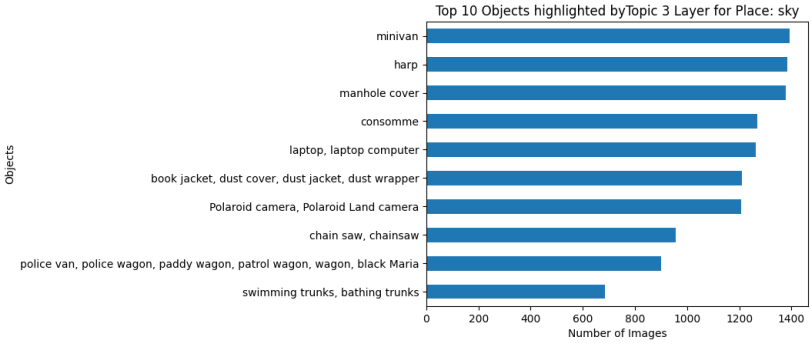


Figure 76: Top 10 occurring objects in Topic 3 layer

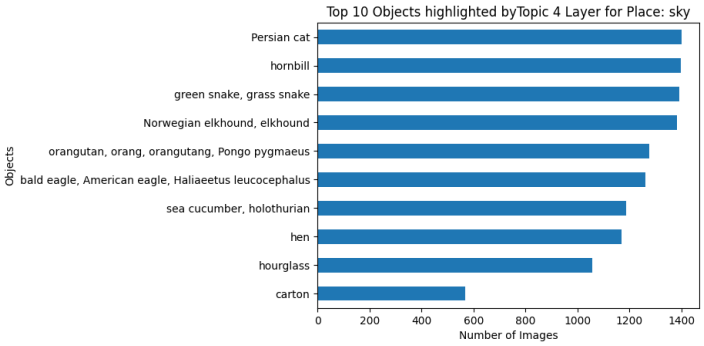


Figure 77: Top 10 occurring objects in Topic 4 layer

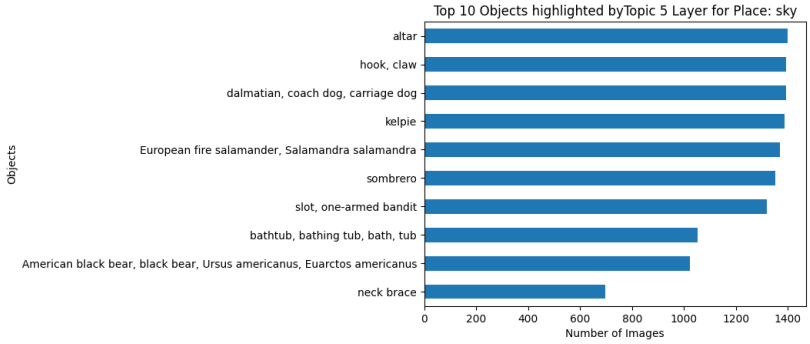


Figure 78: Top 10 occurring objects in Topic 5 layer

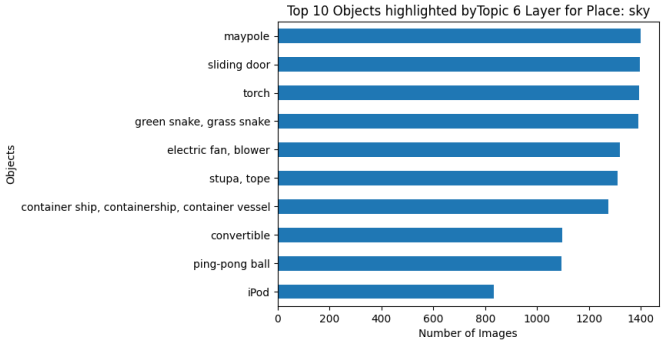


Figure 79: Top 10 occurring objects in Topic 6 layer

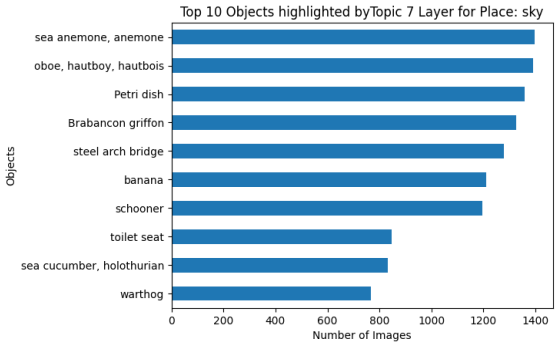


Figure 80: Top 10 occurring objects in Topic 7 layer

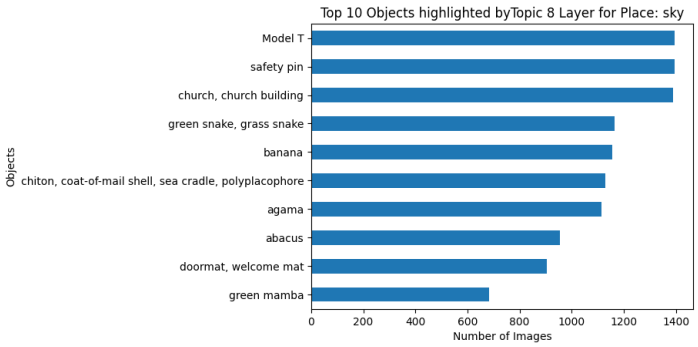


Figure 81: Top 10 occurring objects in Topic 8 layer

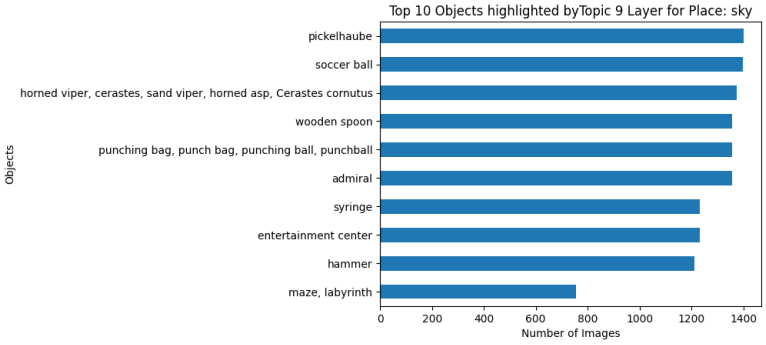


Figure 82: Top 10 occurring objects in Topic 9 layer