Document Classification using Convolutional Neural Networks and Transfer Learning

Mrunal Malekar
malekar.m@northeastern.edu

Shivani Mundle
mundle.s@northeastern.edu

Harshada Sasturkar
sasturkar.h@northeastern.edu

ABSTRACT

Document classification is a task that is performed in many domains on a regular basis, especially in administration. This also constitutes classifying digital images of various documents sent via emails. In this project we aim to use convolutional neural networks to help automate document image classification. As part of our approach we used Convolutional Models and compared its results with pre-trained CNN models used for Transfer Learning. For the Convolutional Neural Network Models two variations were implemented - Time Distributed Layers and Modified Custom Layers (DocNet), whereas for transfer learning VGG16, VGG19 and InceptionV3 were used.Usual document classification models were using Natural language Processing and Text Classification but we planned to classify the documents using layout and structure of the image. The evaluation metrics used for comparison and performance evaluation of the models were accuracy,precision,recall,confusion matrix. Our DocNet CNN was able to achieve 97% accuracy which was the highest among all the other models we implemented.

1. INTRODUCTION

Document classification is a daily part of administration systems that requires officials to scan through thousands of documents. According to statistical reports, there were around 319.6 billion emails sent and received daily in 2021. This trend will be continuing and the projection is that there will be around 376.4 emails generated by 2025. The majority of these emails contain attachments, sent to other individuals or organizations. While it is still manageable for an individual to look through the personal mail attachment, it becomes difficult when the same task is scaled to an organizational level. E.g. Any HR email group must be receiving multiple applications through the candidates with their resumes, previous employment letters, or invoices for reimbursement. Opening each document and storing it in the correct bucket is highly time-consuming and requires extensive human effort.

With the recent advancements in neural networks, it can be possible to have a document image classification system that classifies documents based on their structure and layout. In this

project we will be classifying document images into four categories - letter, form, resume, and invoice. To achieve this we have used two approaches. First is using Custom Convolutional Neural Networks in which we have implemented two model variations - CNN with time distributed layers and global average pooling, and CNN with custom modifications done in the layers (DocNet). In the second approach, we have used Transfer Learning with pre-trained CNN models such as VGG16, VGG19 and InceptionV3 to perform the classification.

Different methods for classifying documents based on their structure have been put forth in the past. The majority of these methods use combinations of specially created local and global features to classify data. These capabilities are appropriate for certain document types and their functionality is significantly impacted by changes to the document corpus. Some methods concentrate on learning and refining features for various document types. The performance of these systems varies when the documents in question change. The main issue with such methods is the necessity for feature set and/or learning mechanism optimization for various document kinds.Consider unrestricted papers, like handwritten documents where a writer is free to write text, notes, or equations, and draw tables or figures, etc., and the problem gets much more difficult . [1] This makes the need to have a classifier which classifies documents using their layout and structure.

In a different study, [2] the method created a codebook of SURF descriptors using features that were taken from a few training images. Then codeword histograms resembling [3] were made. Later, classification was performed using a Random Forest classifier. Even with little training data, the system functioned reasonably.

For image classification there are other supervised machine learning models available like k-Nearest Neighbor, Support Vector Machine and Linear Discriminant Analysis. The reason behind choosing CNN over other supervised classifiers is that CNN is a better feature extractor. It is capable of handling complex data such as images as a direct input and can extract important features from it. CNN is also able to capture the nonlinearity of the data due to its usage of activation functions, unlike other supervised classifiers. Thus, CNN models give higher accuracy. The previous work done on document classification mainly focuses on text analysis using techniques from Natural Language Processing.

For our project we will be focusing on classifying images of documents commonly sent as email attachments. This document image classifier can be used along with the NLP techniques as part of data preprocessing to improve the accuracy of classification.

2. PRELIMINARIES

2.1 DataSet

The Dataset used for this problem statement is the Ryerson Vision Lab Complex Document Information Processing (RVL-CDIP) dataset. The original dataset consists of 16 classes which consist letter,form,email,handwritten, advertisement, scientific report, scientific publication , specification ,file folder ,news article ,budget ,invoice ,presentation ,questionnaire ,resume.From this dataset we decided to build our model on 4 subcategories namely letter, invoice, form, resume. The dataset has been taken from Kaggle and it consists of Training,Validation and Testing Folders.

Each folder consists of 4 categories of sub folders namely letter, invoice, form, resume. Each sub folder in Training consists of 7000 images while in Validation consists of 2000 images and 300 images in Testing.

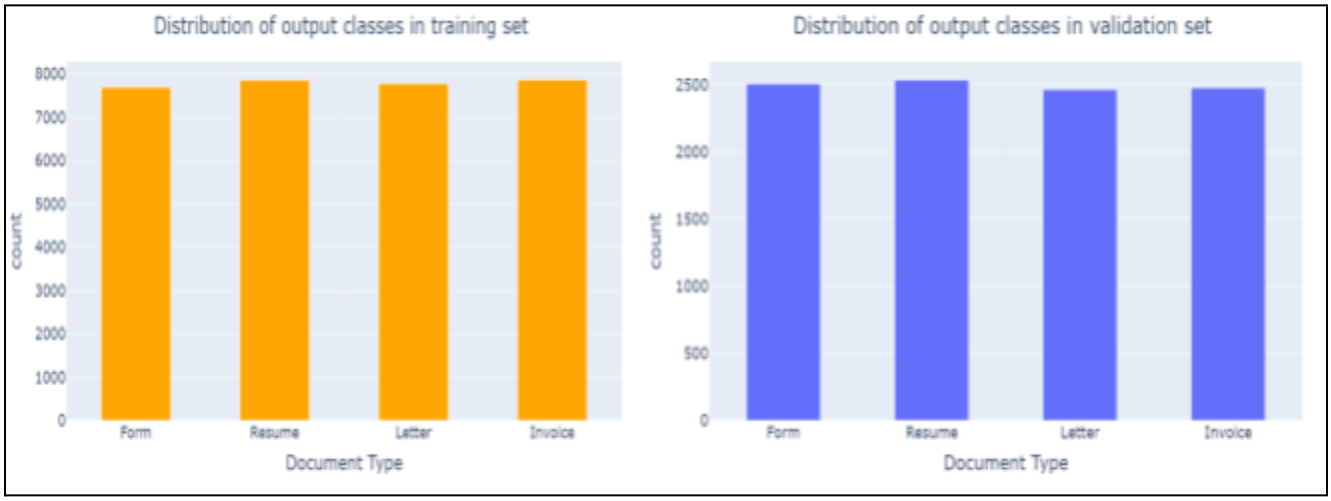


Figure 1: Distribution of classes in Training and Validation Dataset

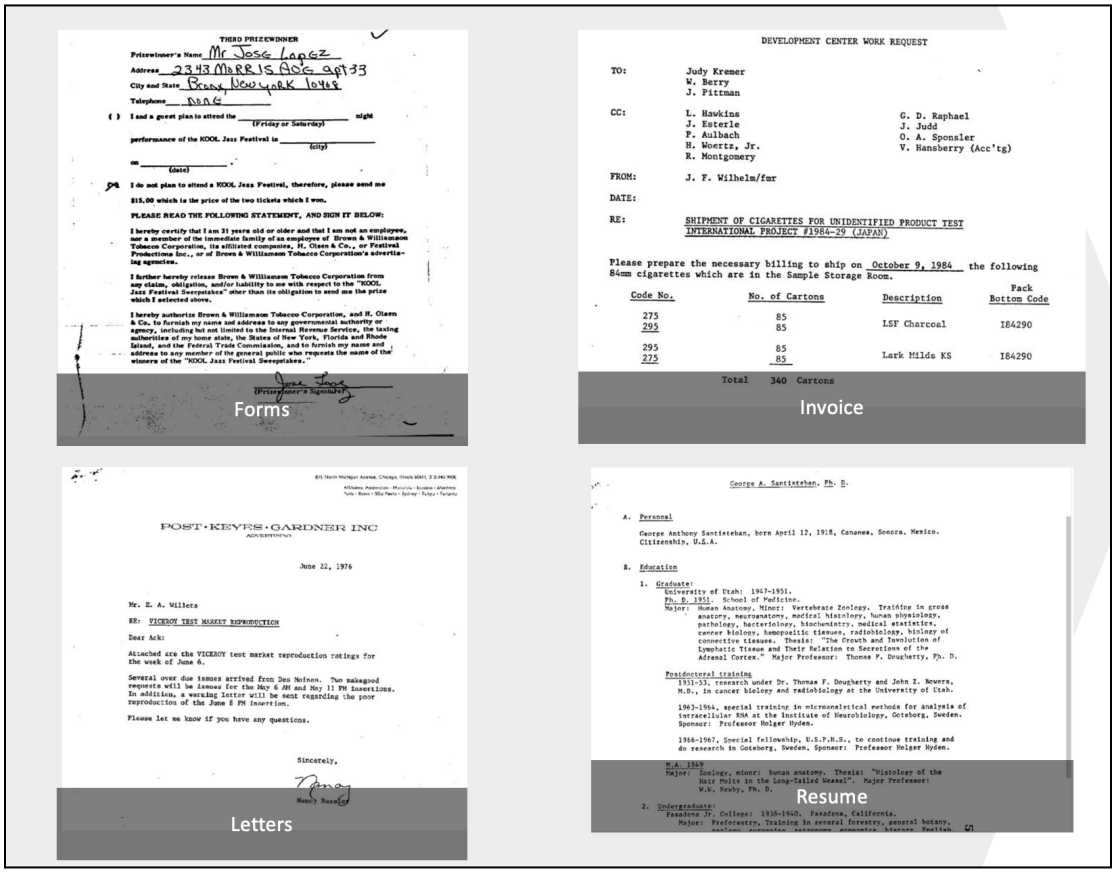


Figure 2: Sample of Original Images

2.2 Initial Setup

The deep learning models used were Convolutional Neural Networks, pre-trained CNN's like RESNET50, VGG16, VGG19, InceptionV3.The deep learning models were trained using Tensorflow. The project was set up and executed on Google Collab Pro with 14.5 GB of RAM and 100 units of GPU. Image Processing was carried out using OpenCV and Matplotlib was used to create graphs.

Table 1: Computational Resources and Setup

Hardware	**Google Collab 14.5 GB RAM 100 units of GPU**
Programming Language	**Python**
Python Libraries	**Pandas, Numpy, Matplotlib, TensorFlow, OpenCV**

2.2 Pre-Processing

2.2.1 Preprocessing for Time Distributed Layers and Custom Convolutional Neural Networks (DocNet)

The default size of each image was 1000 x 764. For time distributed CNN , the images resized to 120,120 and were converted to grayscale. For the customized CNN DocNet, the images were resized to 100 * 100. This size was chosen as we were getting a similar accuracy for both sizes on DocNet. Hence we took the smaller size as it saves computational power.Then they were divided into 4 sub regions(left, right, bottom, top) using the pixels and the region image was resized to 48 x 48 which was followed by normalization.

There was evident noise in the images which was removed using Gaussian filtering. In Gaussian Filtering the average value of surrounding pixels is used to replace the noisy pixel present in the image which is based on Gaussian Distribution.

=

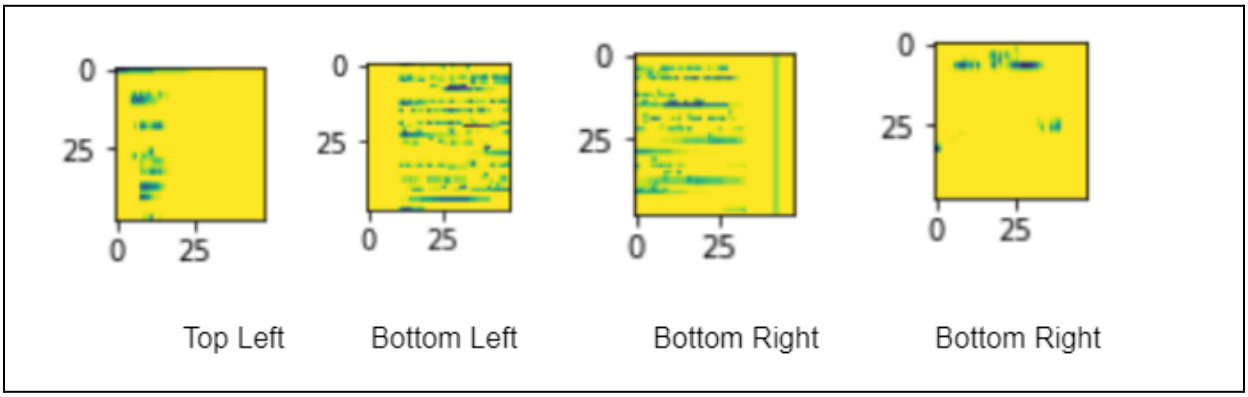


Figure 3: Sample the split image parts.

Gaussian Filter has the advantage such that its Fourier transform is also a Gaussian distribution which is centered around the zero frequency. So, this can help control the effectiveness of the low pass nature of the filter by adjusting its width.

The formula for Guassian filter is given by:

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/(2\sigma^2)}$$

Figure 4: Gaussian Filter Formula

where *x* is the distance from the origin in the horizontal axis, *y* is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

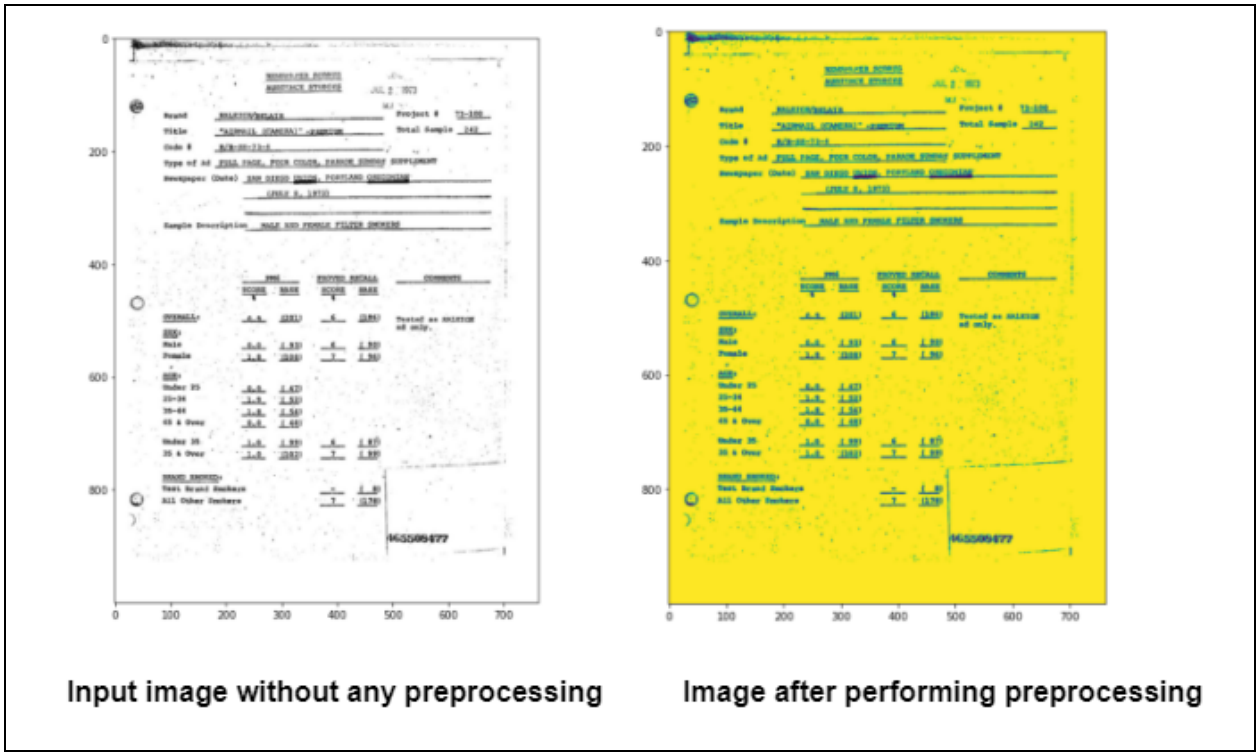


Figure 5: Image before and after preprocessing

2.2.2 Data Augmentation and Labeling

Data was augmented using transformations such as zooming , flipping , cropping , shearing , rotating , etc). Data augmentation helps to create various forms of images by rotating, flipping, zooming. It can help reduce overfitting and act as an approach to over-sample or increase the size of the data.

For transfer learning images were resized to the input shape size as per each pre-trained model which was used namely VGG16, VGG19, InceptionV3.

For outputs, labels assigned to each category (0: form, 1: invoice, 2: letter , 3: resume) were one - hot encoded and concatenated to an array.

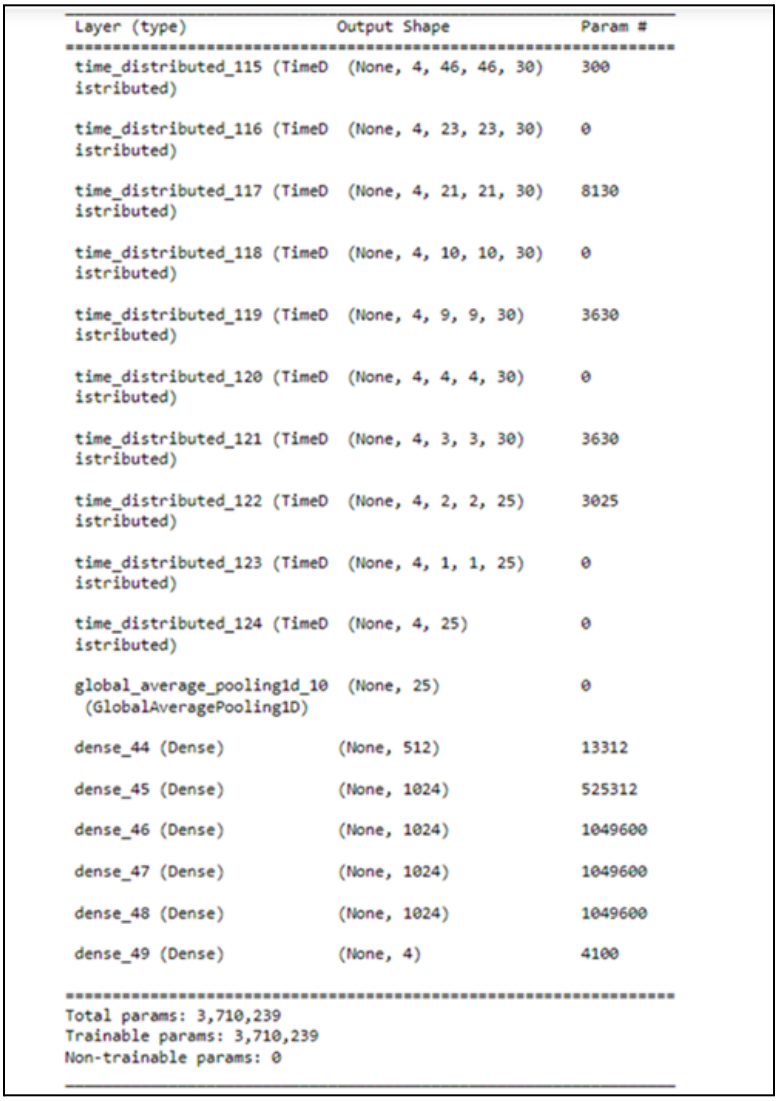
2.3 Methods

2.3.1 Convolutional Neural Networks using Time Distributed Layers

Convolutional The Time Distributed Layer approach divides the images into 4 parts [8]. These four parts are at the top left, top right, bottom left and bottom right. Each corner part is independent. This division helps in extracting similar patterns from every corner of the image.

The images are treated as continuous time-stamped images. The same instance of the weights is used for every part of the image. Doing this gives a better understanding of that image part

independent of relative positioning in the entire image. The output of each time layer is combined and passed to the next layer.Following the time distributed layers, GlobalAveragePooling1D Layer is added in end after all time distributed layers which will be used to find the overall characteristics of the document.
In the end we have added a dense layer to enhance the classification. For the dense layers, we have used the ReLU activation function.Softmax Activation Function is used in the output Layer for multi class Classification.



Layer (type)	Output Shape	Param #
time_distributed_115 (TimeDistributed)	(None, 4, 46, 46, 30)	300
time_distributed_116 (TimeDistributed)	(None, 4, 23, 23, 30)	0
time_distributed_117 (TimeDistributed)	(None, 4, 21, 21, 30)	8130
time_distributed_118 (TimeDistributed)	(None, 4, 10, 10, 30)	0
time_distributed_119 (TimeDistributed)	(None, 4, 9, 9, 30)	3630
time_distributed_120 (TimeDistributed)	(None, 4, 4, 4, 30)	0
time_distributed_121 (TimeDistributed)	(None, 4, 3, 3, 30)	3630
time_distributed_122 (TimeDistributed)	(None, 4, 2, 2, 25)	3025
time_distributed_123 (TimeDistributed)	(None, 4, 1, 1, 25)	0
time_distributed_124 (TimeDistributed)	(None, 4, 25)	0
global_average_pooling1d_10 (GlobalAveragePooling1D)	(None, 25)	0
dense_44 (Dense)	(None, 512)	13312
dense_45 (Dense)	(None, 1024)	525312
dense_46 (Dense)	(None, 1024)	1049600
dense_47 (Dense)	(None, 1024)	1049600
dense_48 (Dense)	(None, 1024)	1049600
dense_49 (Dense)	(None, 4)	4100

Total params: 3,710,239
Trainable params: 3,710,239
Non-trainable params: 0

Figure 6: Architecture of the Time Distributed Layers CNN.

After running different combinations of dense layer sizes, this architecture gave us the best result. As we can observe we need 6 dense layers to correctly classify documents after feature extraction using the convolutional neural network.

2.3.2 Custom Convolutional Neural Networks

Convolutional Neural Network is a specialized artificial neural network to analyze images. The convolutional neural network uses the convolutional, max pooling and finally dense layers to classify the images.

Convolutional Layers: Convolutional layer uses 2D convolutional layers to combine learned features with input data. This network is therefore well suited to the processing of two-dimensional images.In comparison to other image classification techniques, CNNs require very little preparation.As a result, applying the filter to an input several times allows you to detect the characteristics of the input. Filters, kernel size, activation, and input shape are all parameters used in the convolutional layer.The activation function in all convolutional layers was ReLu activation. The rectified linear activation function, or ReLU, is a piecewise linear function that will output the input directly if it is positive and zero otherwise.

MaxPool Layers:After the convolution layers, maxpool layers are used. As a result, it aids in selecting the most significant element from the feature map's region. As a result, it aids in the selection of the maximum element from the region of the feature map.

Flattening and Dense Layers:The data is then flattened using a flatten Layer, which converts it into a one-dimensional array that can be passed on to the next layer. We flatten the output of the convolutional layers to create a single long feature vector. It is also linked to the final classification model, which is referred to as a fully-connected layer.After the flatten layer, the Dense Layer is used. The regular deeply connected neural network layer is the dense layer. It is the most popular and widely used layer.ReLU was used in the Dense layers to introduce non linearity in image data.Softmax Activation was used in the Output Layer with cross entropy Loss.

Batch Normalisation: Instead of doing the normalization in the raw data, batch normalization is done between the layers of a neural network. It is done in mini-batches rather than the entire data set.Using Batch Norm ensures that the mean and standard deviation of the layer inputs, beta and gamma, will always be the same. As a result, the amount of change in the distribution of layer input is reduced. The deeper layers have a more solid understanding of what the input values will be, which aids in the learning process.

We used batch normalization as a technique to deal with the overfitting in our model.The data distribution of the model experiences noise each time since it is computed across mini-batches rather than the whole data set. This can serve as a regularizer, preventing overfitting and enhancing learning.

We have tried many combinations of the number of layers, convolutional filter sizes,number of max pooling layers, batch normalization layers. We evaluated the test and validation accuracies for all these models and then came up with our final CNN model which gave us the best accuracy of 97%.

Table 2: Model Configurations and Test Accuracies

Layers	Kernel Size used in Convolutional Layer	Max Pooling Filter Sizes	Accuracy
Convolutional : 4 Pooling:3 Normalization : 3 Dense : 4	9*9*50 5*5*50 3*3*35	3*3	97.36
Convolutional : 3 Pooling : 3 Normalization : 3 Dense : 3	9*9*50 5*5*50 3*3*35	5*5 2*2	81.76
Convolutional : 5 Pooling : 3 Normalization : 3 Dense : 1	11*11*50 3*3*35 1*1*30 1*1*20	3*3	84.34

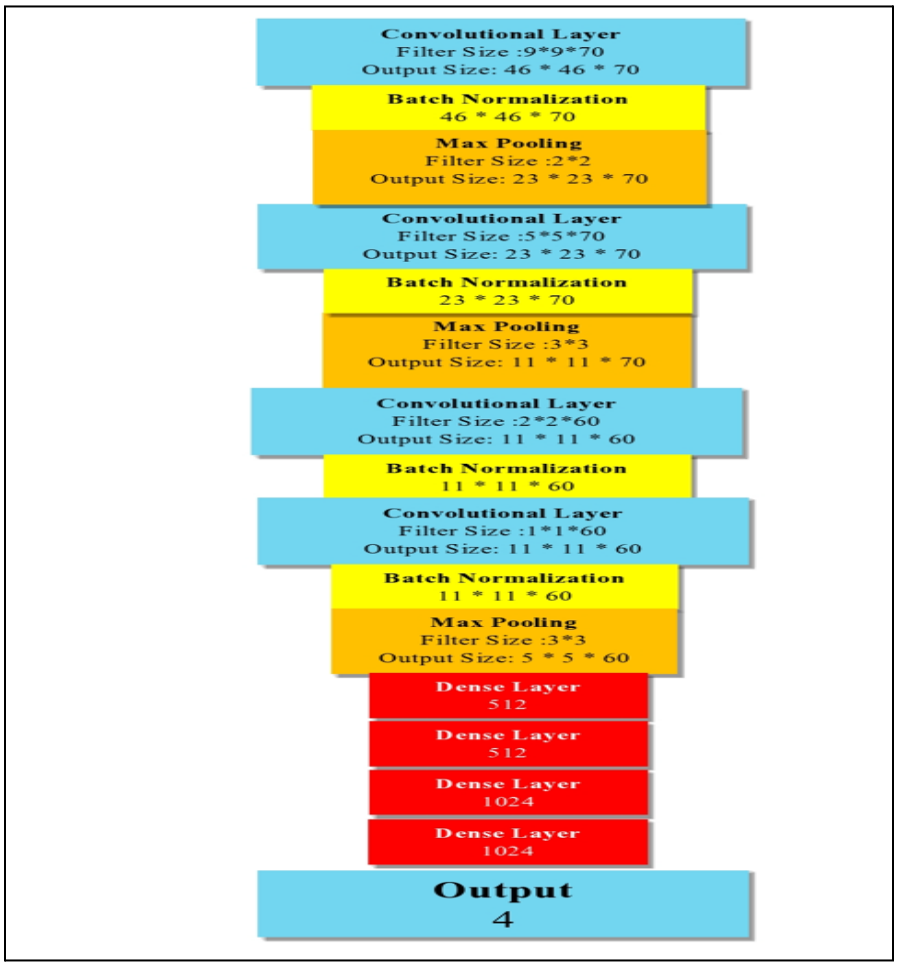


Figure 7: Architecture of the customized CNN with best results (DocNet)

2.3.3 Transfer Learning Models

Transfer learning is a technique where a model developed for a task is reused as the starting point for a model on a second task. Transfer learning is an optimization that allows rapid progress or improved performance when modeling the second task.

In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.For example, a model trained on a large dataset of cats picture would contain learned features like the edges which can be used and transferred for your dataset.Pre-trained models have lots of advantages. They help save time and resources as pre-trained models are trained by someone else and trained on a large dataset to solve a similar dataset. In our approach we used fine tuning where we unfreezed the layers of the pre trained model took the output of its last layers, added some extra dense layers with ReLU Activation, drop out layers and output layer with width 4 for our output classes and re -trained the entire model on our images data with a very low learning rate . We used Adam Optimizer and categorical cross entropy Loss. The batch size was set to 32 and 5 epochs were used. The learning rate was set to 0.0001 .

We used ReduceLROnPlateau as a scheduling technique that was used so that it decreases the learning rate when the metric specified which was validation loss stops improving for longer than the patience number allows. We used a patience of 2.ModelCheckPoints was used to save the best model or model's weights when it achieves its best performance with respect to training accuracy. Below we have summarized the pre-trained CNN's models that were used along with its architecture and model structure.

VGG16: The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer. VGG16 takes input tensor size as 224, 244 with 3 RGB channels.

Figure 8: VGG-16 Pre - Trained Model Architecture

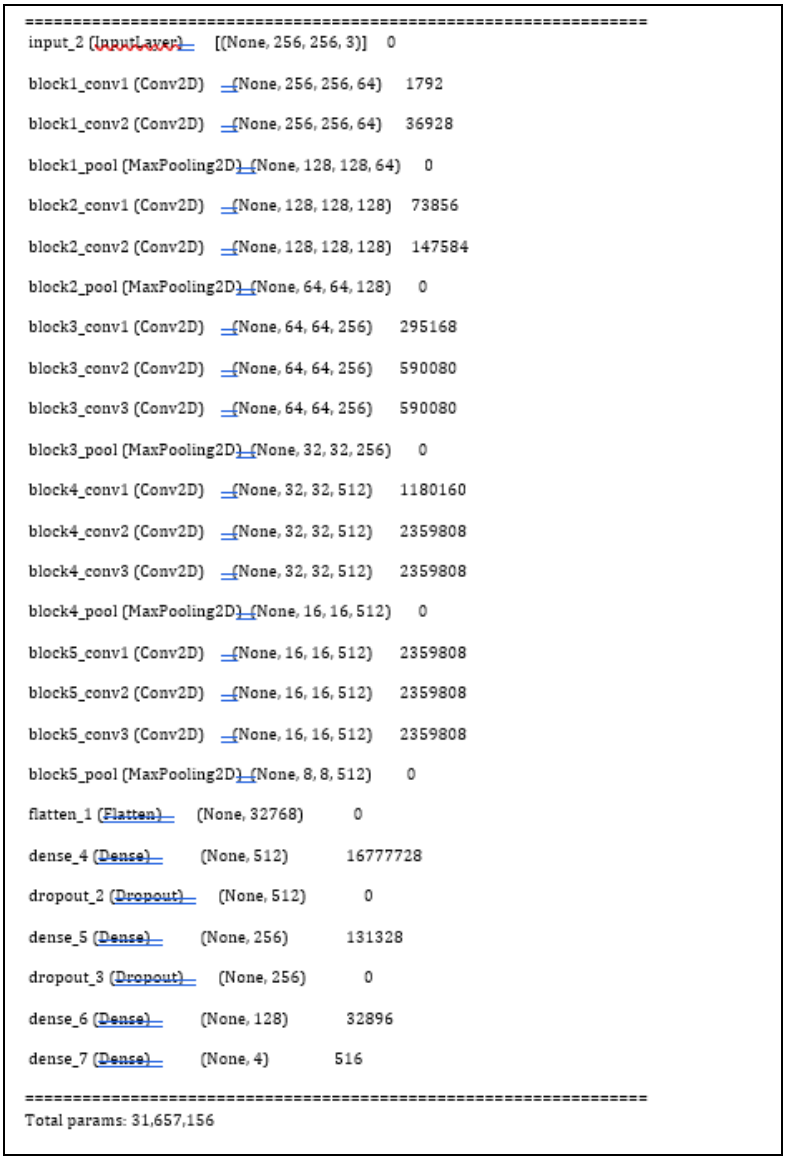


Figure 9: VGG 16 - Model Structure

VGG19 :- VGG19 is a variant of the VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layers, 5 MaxPool layers and 1 SoftMax layer). Fixed size of (224 * 224) RGB image was given as input to this network which means that the matrix was of shape (224,224,3).The only preprocessing that was done is that they subtracted the mean RGB value from each pixel, computed over the whole training set.

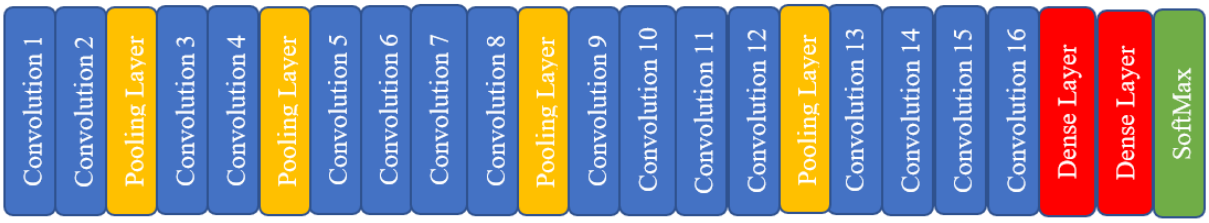


Figure 10: VGG19 Model Architecture

VGG19 Architecture used in this along with a number of parameters is summarized below in its model summary.

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 256, 256, 3)]	0
block1_conv1 (Conv2D)	(None, 256, 256, 64)	1792
block1_conv2 (Conv2D)	(None, 256, 256, 64)	36928
block1_pool (MaxPooling2D)	(None, 128, 128, 64)	0
block2_conv1 (Conv2D)	(None, 128, 128, 128)	73856
block2_conv2 (Conv2D)	(None, 128, 128, 128)	147584
block2_pool (MaxPooling2D)	(None, 64, 64, 128)	0
block3_conv1 (Conv2D)	(None, 64, 64, 256)	295168
block3_conv2 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv3 (Conv2D)	(None, 64, 64, 256)	590080
block3_conv4 (Conv2D)	(None, 64, 64, 256)	590080
block3_pool (MaxPooling2D)	(None, 32, 32, 256)	0
block4_conv1 (Conv2D)	(None, 32, 32, 512)	1180160
block4_conv2 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv3 (Conv2D)	(None, 32, 32, 512)	2359808
block4_conv4 (Conv2D)	(None, 32, 32, 512)	2359808
block4_pool (MaxPooling2D)	(None, 16, 16, 512)	0
block5_conv1 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv2 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv3 (Conv2D)	(None, 16, 16, 512)	2359808
block5_conv4 (Conv2D)	(None, 16, 16, 512)	2359808
block5_pool (MaxPooling2D)	(None, 8, 8, 512)	0
flatten (Flatten)	(None, 32768)	0
dense (Dense)	(None, 512)	16777728
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
dense_3 (Dense)	(None, 4)	516

Total params: 36,966,852
Trainable params: 36,966,852
Non-trainable params: 0

Figure 11: VGG19 Model Architecture

InceptionV3- Inception-v3 is a convolutional neural network that is 48 layers deep. The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average

pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using Softmax.

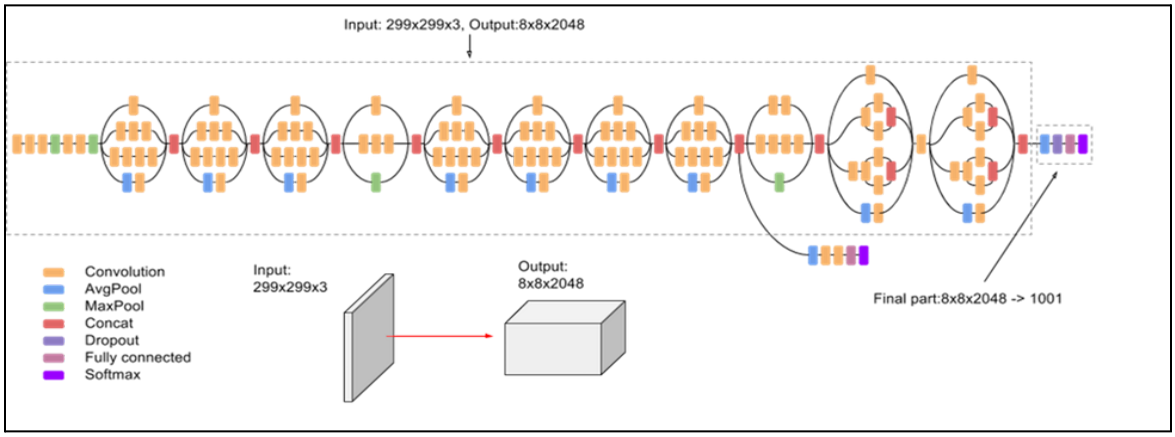


Figure 12: InceptionV3 Model Architecture
(https://paperswithcode.com/method/inception-v3)

3. RESULTS

The main aim of the project was to correctly classify the images into four categories namely letter,resume,invoice,form using Convolutional Neural Networks and Transfer Learning using VGG19,VGG16 and InceptionV3. The proposed classification system can help classify document type according to structure.

3.1 Main Results

Four different pre-trained transfer learning models were used and following metrics were used to evaluate the performance of the model:

a) Training and Validation Loss
b) Training and Validation Accuracy
c) Recall
d) Precision
e) Confusion Matrix (4 class category)

3.1.1 Convolutional Neural Networks Results

For the Time Distributed Neural Network , we performed a preprocessing of splitting the image into 4 parts and feeding it as an input to the time distributed layer. After performing the pre-processing, the model was trained on the training dataset in Google Colab. The model got an accuracy of 81%. The accuracy did not increase beyond 81% even with different parameters. We believe that the notion of time did not work very well for a static document and hence we used the entire image for our custom CNN.

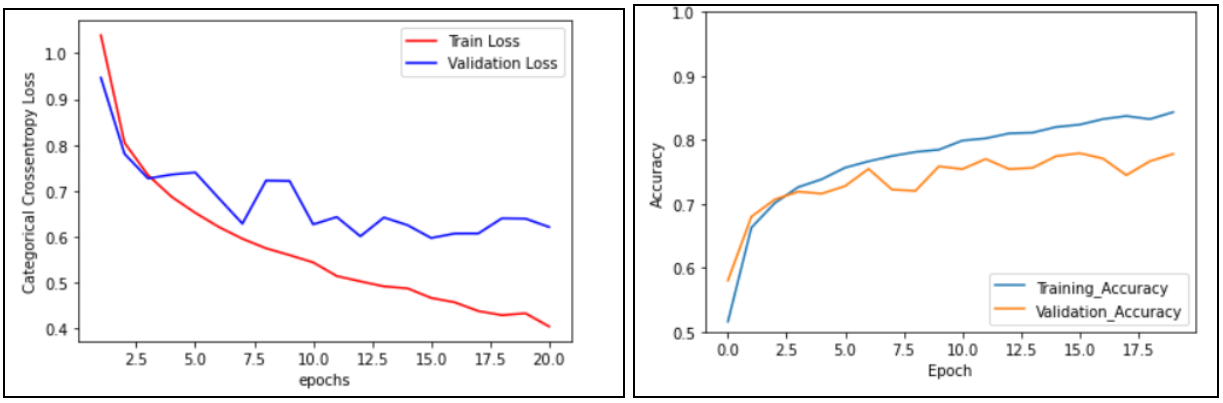


Figure 13: Epochs vs Training and Validation Loss/Accuracy for Time Distributed CNN

For the custom neural network (DocNet), we used multiple combinations of batch, epoch, kernel sizes. After performing the pre-processing, the model was trained on the training dataset in Google Colab. After multiple attempts we got a model which gave us 97 % accuracy. We were initially not crossing an accuracy of 84%. After adding Batch Normalization and DropOut Layer the accuracy got a little better. After that we performed train data shuffling which gave us a really good accuracy of over 94%.

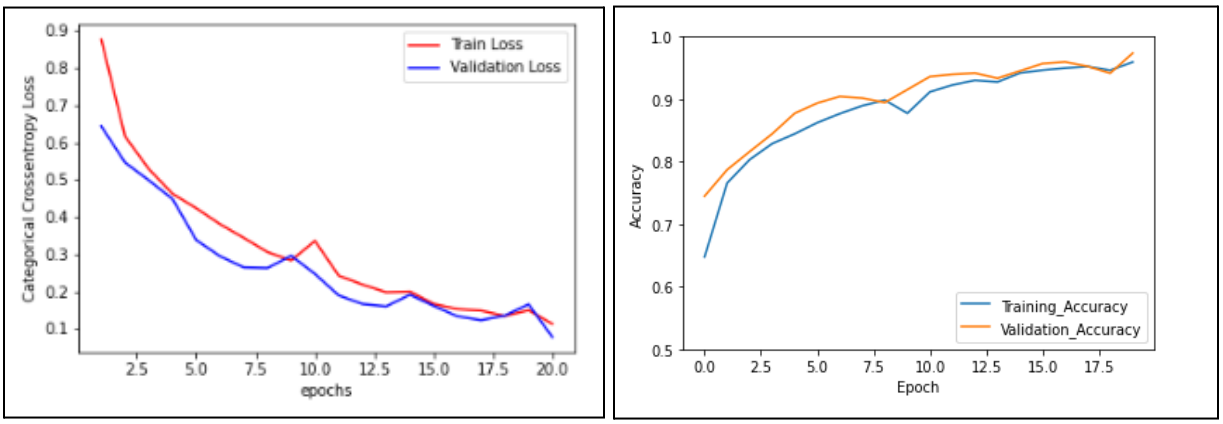


Figure 14: Epochs vs Training and Validation Loss/Accuracy for DocNet

3.1.2 Transfer Learning Results

We plotted training and validation accuracy and loss curves for different values of epoch sizes and drop out Layers. We trained three different pretrained models namely VGG19,VGG16 and Inception V3 using different batch sizes,epochs sizes and hidden layers. Using configuration of 5 epoch sizes and two drop out layers with 256 x 256 input shape size the following plots were obtained with respect to training and validation accuracies and losses. The model was trained on Google collab GPU with batch size 32 and learning rate=0.0001 using Adam Optimizer.

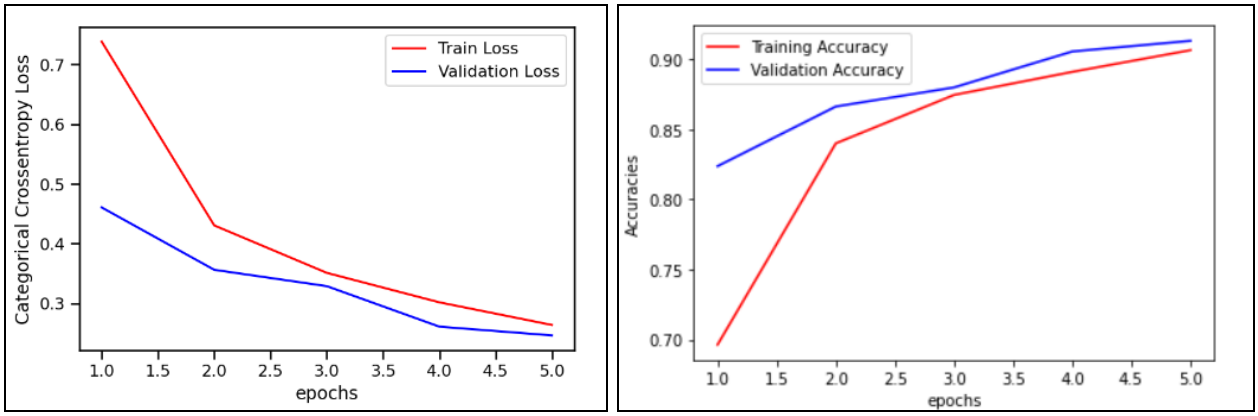


Figure 15: Epochs vs Training and Validation Loss/Accuracy for VGG19

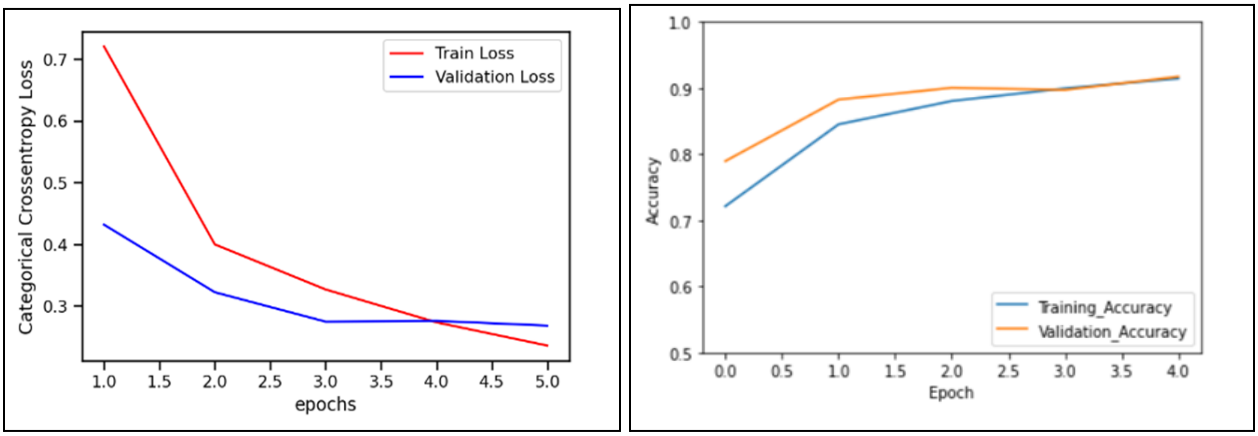


Figure 16: Epochs vs Training and Validation Loss/Accuracy for VGG16

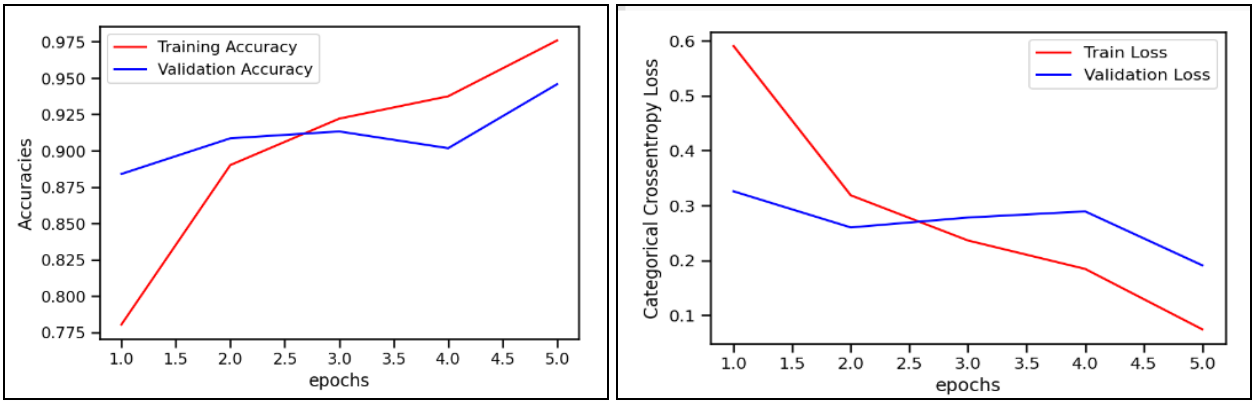


Figure 17: Epochs vs Training and Validation Loss/Accuracy for InceptionV3

Confusion Matrix:

As the data distribution in each class is balanced, we felt accuracy will be a good metric for evaluating the model's performance .We decided to use the confusion matrix to visualize the number of true positives, false positives, true negatives, false negatives.The confusion matrix is

a table which has 16 combinations of true and predicted values. It is important evaluation metric to measure performance on model.

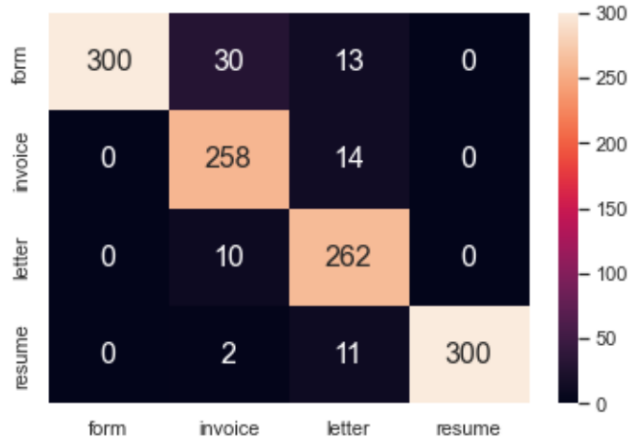


Figure 19: Confusion Matrix for Customized CNN (DocNet)

The above shows confusion matrix that was obtained using the best performing DocNet CNN model for classification of testing images in four categories (0: form, 1: invoice, 2: letter , 3: resume).
The overall results showed that model's performed marginally better after application of dropout layers and reducing the epoch sizes.The higher the accuracy, the better is the model at predicting the correct class category for the document.

3.1.3 Comparison of Results

	Models	Testing Accuracies	Recall	Precision
0	VGG19	0.9432	0.9356	0.9034
1	InceptionV3	0.9032	0.9011	0.8900
2	VGG16	0.8832	0.8889	0.8960

Figure 20: Comparison of Evaluation Metrics of various pre-trained CNNs on Testing Data

	Models	Testing Accuracies	Recall	Precision
0	Time Distributed Convolutional Neural Network	0.81390	0.7703	0.8482
1	Customized Convolutional Neural Network (DocNet)	0.97363	0.9668	0.9798

Figure 21: Comparison of Evaluation Metrics of custom CNNs on Testing Data.

3.2 Supplementary Results

We chose the Relu Activation function for the Dense Layers and Global AveragePooling in the Convolutional Neural Network Model which used Time Distributed Layers. Relu was chosen as it increases non-linearity in Images do not suffer from vanishing gradients and converge faster. Softmax Activation was used in the Output Layer. The hyperparameters chosen for the Time Distributed Convolutional Neural network involve 120 * 120 image size. A batch size of 64 for 20 epochs was used. Early stopping was also implemented to avoid overfitting. ReLu is chosen as the activation function for hidden layers and Softmax is used for the output layer. The 'adam' optimizer along with categorical_crossentropy is used.

```
=================================
Total params: 5,255,172
Trainable params: 5,255,172
Non-trainable params: 0
_________________________________
```

Figure 22: Parameter Summary for Time Distributed CNN.

The hyperparameters chosen for the custom Convolutional Neural network (DocNet) involve 100* 100 image size. A batch size of 64 for 20 epochs was used. Early stopping was also implemented to avoid overfitting. ReLu is chosen as the activation function for hidden layers and Softmax is used for the output layer. The 'adam' optimizer along with categorical_crossentropy is used. Additionally 2 dropout layers are used along with the 4 convolutional layers. The parameter summary is shown below.

```
=================================
Total params: 2,925,674
Trainable params: 2,925,154
Non-trainable params: 520
_________________________________
```

Figure 23: Parameter Summary for custom CNN DocNet.

The hyperparameters chosen for Transfer Learning involve 256 x 256 image size with batch size of 32 and epochs of 15. The learning rate is chosen as 0.001 .Softmax Activation was used for the Output Layer and Linear Activation was used in the Hidden Layers. We have also added two Drop Out Layers of 0.5 after each Dense layer in the final layers of the Transfer Learning

model. The three Dense layers added have 512 ,256 and 128 neuron size each. Stochastic Gradient Descent(SGD) and categorical cross entropy loss was used for multi class classification. Cross Entropy works by trying to minimize the distance between actual and predicted values. We saw that initially with two Dense Layers that were added on top of the pretrained model and no dropouts there was significant overfitting in the model's performance. Later with increase in neurons, addition of dropouts and reducing the epoch sizes models showed that overfitting reduced and we were able to generate better validation and training accuracy/loss curves. The best performing transfer learning model and its parameter summary can be seen below.

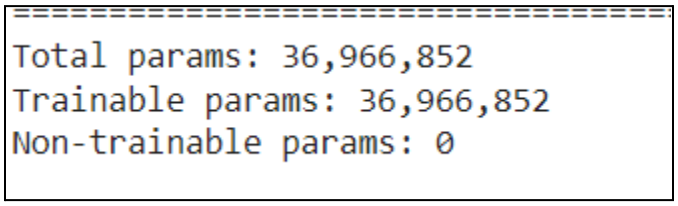


Figure 24: **Parameter Summary for VGG19.**

4. DISCUSSION

In both the approaches of CNN and transfer learning, we got test accuracy scores above 80%. For the DocNet CNN model we were able to achieve an accuracy of 97% which was the highest of all other models. The transfer learning models came next with the accuracies of 94%, 90% and 88% for VGG19, InceptionV3 and VGG16 respectively. For the time distributed CNN we got 81% accuracy. It can be seen from the results table that we got good precision and recall values for all the models. We made sure to check for and reduce overfitting as much as possible as evident from the train and validation loss graphs which have considerably less gap between them. A similar project that we had referred to used deep CNN for document classification and obtained an accuracy of 77.6% [1]. Another project which did automatic document classification using CNN reached an accuracy of 90% [4]. Compared to that we achieved higher accuracies for our models. Thus, we were able to successfully perform document image classification with good results.

In future we hope to work on a larger and more diverse dataset to analyze how well our models work and whether the performance could be improved. This problem statement can be further expanded to use NLP techniques after image classification to gather further information from these documents. This can also be a helpful addition to the Optical Character Recognition(OCR) pipeline which is used to extract the raw text from images. Since it is difficult to build a generalized OCR for all types of documents, the image classifier can be attached prior to the OCR pipeline to preprocess different documents.

5. CONCLUSION

In this project, we performed document image classification using convolutional neural network models and transfer learning. We implemented our own two versions of CNN models - one with time distributed layers and another with customly modified convolution layers(DocNet). For these CNN models we got accuracies of 97% and 81% respectively. For the transfer learning part we implemented three pre-trained CNN models, VGG16, VGG19 and InceptionV3. Out of these VGG19 gave the highest accuracy of 94% followed by InceptionV3 with 90% and VGG16 with 88%. From the results we can conclude that our DocNet CNN model worked better than the pre-trained CNN models.

6. REFERENCES

1. Deepdocclassifier: Document classification with deep Convolutional Neural Network by Muhammad Zeshan Afzal, Samuele Capobianco, Muhammad Imran Malik, Simone Marinai, Thomas M. Breuel, Andreas Dengel, and Marcus Liwicki
2. Convolutional Neural Networks for Document Image Classification by Le Kang. Jayant Kumar, P eng Ye, Yi Li, and David Doermann
3. Learning document structure for retrieval and classification by J. Kumar, P. Ye, and D. S. Doermann
4. Automatic Document Classification Using Convolutional Neural Network by Xingping Sun, Yibing Li, Hongwei Kang, and Yong Shen
5. Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval by Adam W. Harley, Alex Ufkes, and Konstantinos G. Derpanis
6. https://medium.com/analytics-vidhya/how-i-built-a-document-classification-system-using-deep-convolutional-neural-networks-e1d9a83cbabd
7. https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/
8. https://www.kaggle.com/code/kaledhoshme/documents-classification-using-cnn
9. https://github.com/kaledhoshme123/Documents-Classification-Using-CNN