Document Classification Using Deep Learning

DA 526 : *Image Processing with Machine Learning*

Git Repo : https://github.com/LoneWolfDiesButThePackSurvives/DA526_IPML

Submitted By :-

Atul Bhagat 224156004

Rishabh Tomar 224161007

Instructor - Dr. Debanga Raj Neog

16 May 2023

Even Semester

Academic Year 20**22-23**

Abstract

This project explores image classification using deep learning techniques applied to the RVL-CDIP Database. The study aims to develop an accurate classification system by preprocessing images, constructing deep neural network architectures, and training the models using the dataset. Different model architectures like AlexNet, VGG-19, Res-Net etc. were developed and evaluated using tensor flow framework in python. Evaluation metrics like F1 score and accuracy were used to assess the model's performance. The findings highlights the data hungriness of these models and acknowledges that deep learning models **might not capture** all the essential **contextual information** required for document classification tasks. Consequently, their applicability could be restricted in specific scenarios.

Also, apart from just training different models, we tried to extract the hidden layer activations and tried explaining the classification process, moving a bit towards explainable-AI.

1. Introduction

With exponential increase in data generation, there is a need of sophisticated systems that can churn through this humongous data and could cluster or label this data so that it could be used for the downstream tasks. Recent years have proven to be very successful for the deep learning solutions and in the computer vision bubble, Convolution Neural Networks (CNNs) have proved their worth over and over again. Although it must be noted that many transformers based solution with attention nowadays seem to be the new silver bullet, they aren't in the scope of this study but they fully deserve their mention here.

Computer Vision could be summarised as extracting meaningful information from the digital images in an automated fashion. It includes fields like Image Classification, Object Identification, Detection and Localisation, Segmentation, Pose Estimation etc. We would deal with the field of image classification and even in the field of image classification, this study would focus on "Document Classification" rather than object classifications like Fashion-MNIST database. Any image inherently has a spatial structure based on which we determine its class, like two loops connected together form an 8 or Infinity, depending on the orientation. If we could device an algorithm that could extract these sort of spatial relations from the images that is rotation and scaling invariant, it would classify these images perfectly. However, when we consider a document, not only it does have a structure but it also has an added context i.e the textual information inside the document. So based on this there could be two different approaches for this classification, i.e Optical Character Recognition (OCR) based solutions, which would try to extract the textual information from the given documents, thus having the access to the contextual information of the documents. Other approach is to use CNN based solutions where it would try to classify based on the spatial information only. There could be ensemble approach which utilises both of these information to make the final decision.

In this project, we'll be classifying the documents based on the spatial information only. The motivation behind this approach is that us humans could easily categorise different documents, with good accuracy, with spatial information only. Also, it is important to note that OCR based solutions are very time consuming, and spatial based solutions are quite quicker and efficient. OC R based ensemble solutions were are also tried on a smaller dataset for this project, but since it is very time consuming and doesn't guarantee correct textual information extraction, this part was excluded from this report.

So in summary, this project aims at developing different CNN architectures on the RVL CDIP Dataset and analysis and evaluation of each model is conducted in this report.

2. DataSet

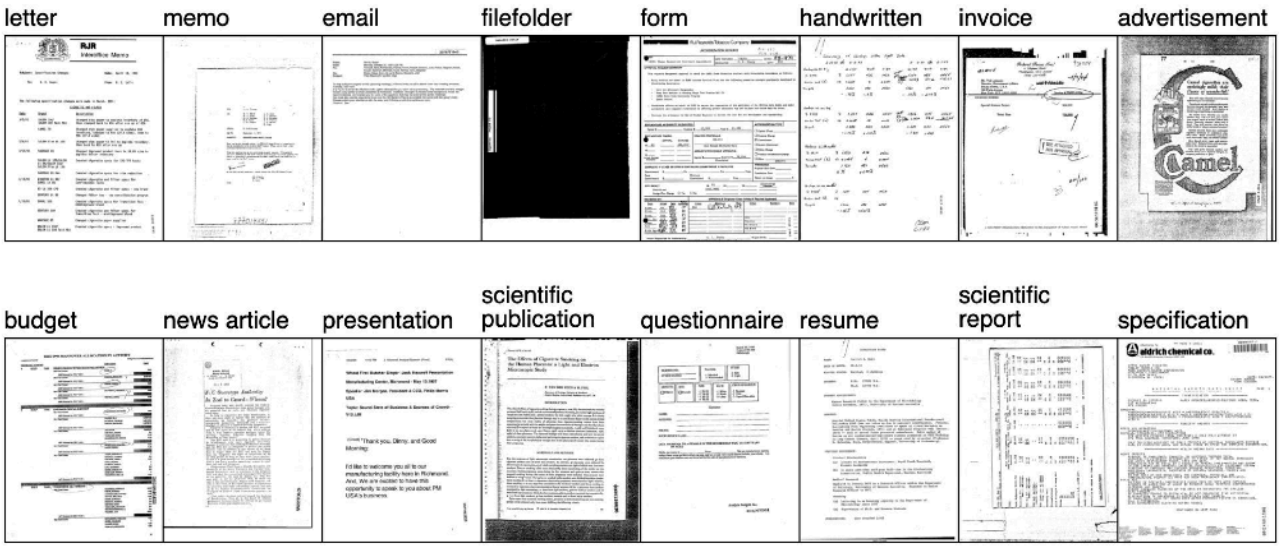


Figure 1 - Sample Images in RVL-CDIP Dataset

The dataset used for this project is **RVL-CDIP**. The RVL-CDIP dataset consists of scanned document images belonging to **16 classes** such as letter, form, email, resume, memo, etc. The dataset has **320,000 training**, 40,000 validation and 40,000 test images. This is a very commonly used dataset for classification tasks. Computer vision community has tried different approaches such as using **localised CNNs** that focuses on different areas of the documents like header, footer, centre columns etc. in the document and classify based on this information. Other approaches were using **transfer learning**, and it was learnt the features learned on a generalised training database are robust for document classification as well. Then there are attention based transformers model that outperform all these approaches by a significant margin.

3. Methodology

3a) Data Loading and Augmentation

We use *ImageDataGenerator* class provided by the tensor flow's Keras module. This works in two steps i.e first we create a data generator that defines the augmentations needed for our training and then using this data generator a data iterator is created which pulls the data from the desired path and loads data in batches. We use data augmentation since it helps prevent model overfitting, which usually happens when training large models with a scarce dataset. The Augmentations used for this project were as follows :

- *Scaling* - all the pixel values are compressed to a range of (0-1)
- *Resizing* - input of 1024 x 1024 is converted to 256 x 256
- *Image Rotation* - Each image is randomly rotated by [-10 +10] degrees
- *Width Shift* - Each image is shifted along its width by a random parameter between [0, 0.3]
- *Height Shift* - Each image is shifted along its height by a random parameter between [0, 0.3]
- *Shear Range* - Each image is also sheared with a small amount randomly selected between [0, 0.1]
- *Horizontal Flip* - Each image is also horizontally flipped randomly
- *Fill mode* - "Nearest" fill mode was used in all the cases

3b) Hyper-parameter Tuning and Model Selection

Hyper-parameter tuning and model selection are crucial steps in any machine learning task. In our project, the main hyper-parameters that we tweaked were the optimiser, learning rate, and dropout rate. For few architectures grid search from keras was utilised to get the optimum parameters, but because of the high time consumption, a trial and error and utilising industry standard ranges for these parameters was adopted. This resulted in having more than 70+ trained models and model selection was done based on the best

3c) Hidden Layer Activation extraction

Apart from just training a classifier, a comparison across all the model architectures (best models) was done based on the hidden layer activations. All the hidden layer weights were extracted and passed through with same image, to get an idea of what the network was trying to learn and why was it failing in few cases. Using Matplot lib, each hidden layer activation for all the filters were visualised and stored as a JPEG file.

3d) Classification Report

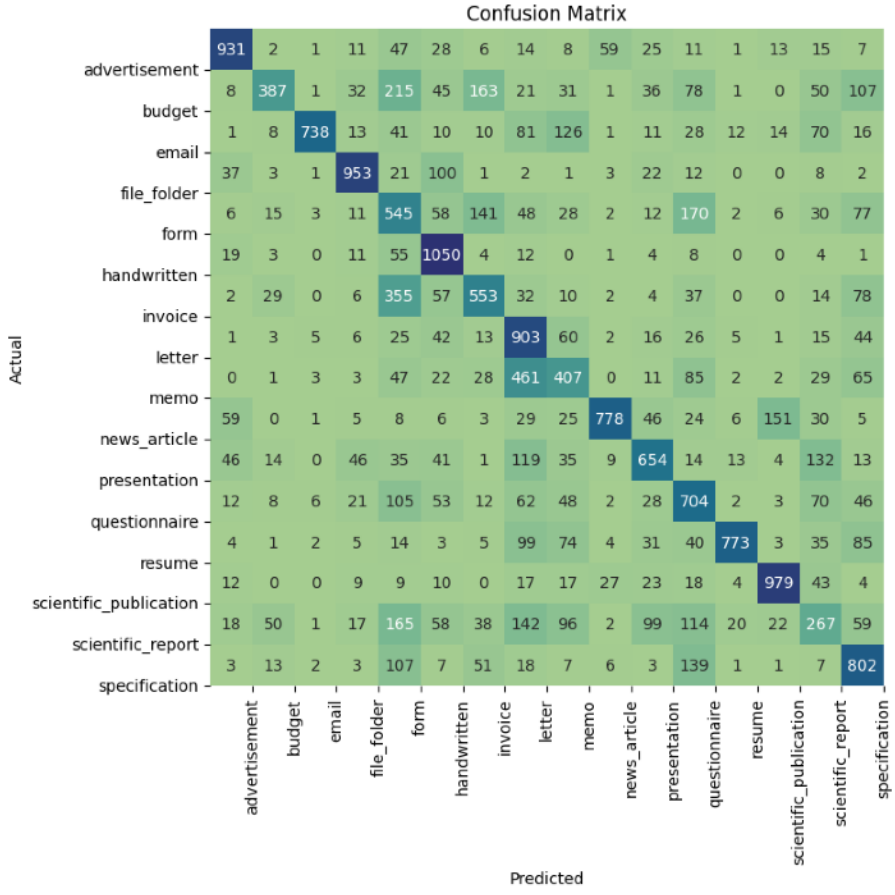
The models were evaluated based on their F1 Score and Classification Accuracy. To assess the performance of each model, a Confusion Matrix was created using the matplotlib and sci-kit learn packages. The Confusion Matrix provides insights into the true positive, true negative, false positive, and false negative predictions.

4. Results

4.1) Basic CNN

Classification Report

	precision	recall	f1-score	support
advertisement	0.8033	0.7897	0.7964	1179
budget	0.7207	0.3291	0.4518	1176
email	0.9660	0.6254	0.7593	1180
file_folder	0.8273	0.8173	0.8223	1166
form	0.3038	0.4723	0.3697	1154
handwritten	0.6604	0.8959	0.7603	1172
invoice	0.5374	0.4690	0.5009	1179
letter	0.4383	0.7738	0.5597	1167
memo	0.4183	0.3491	0.3806	1166
news_article	0.8654	0.6616	0.7499	1176
presentation	0.6380	0.5561	0.5943	1176
questionnaire	0.4668	0.5956	0.5234	1182
resume	0.9181	0.6562	0.7653	1178
scientific_publication	0.8165	0.8353	0.8258	1172
scientific_report	0.3260	0.2286	0.2687	1168
specification	0.5684	0.6855	0.6215	1170
accuracy			0.6089	18761
macro avg	0.6422	0.6088	0.6094	18761
weighted avg	0.6429	0.6089	0.6098	18761



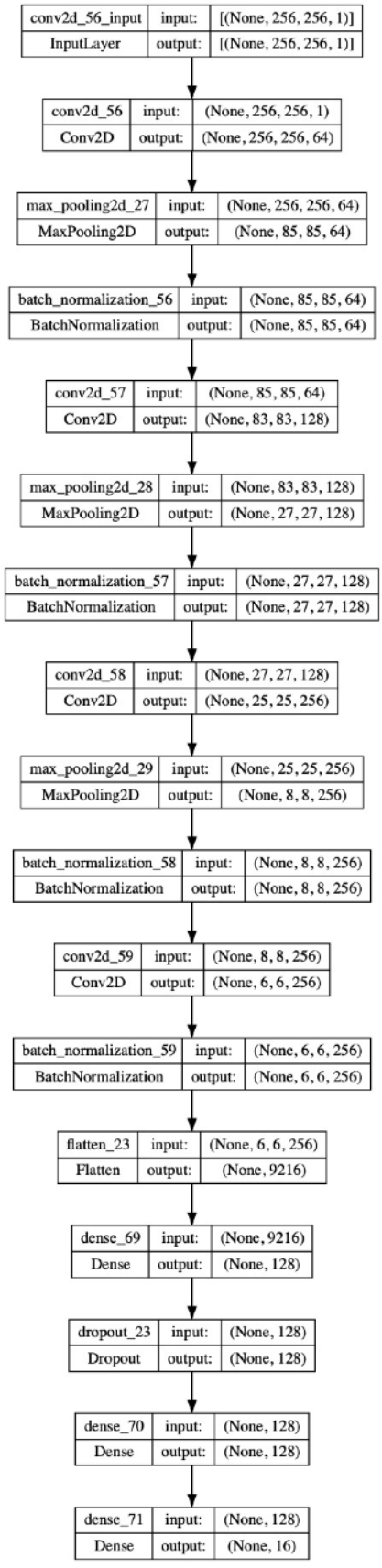
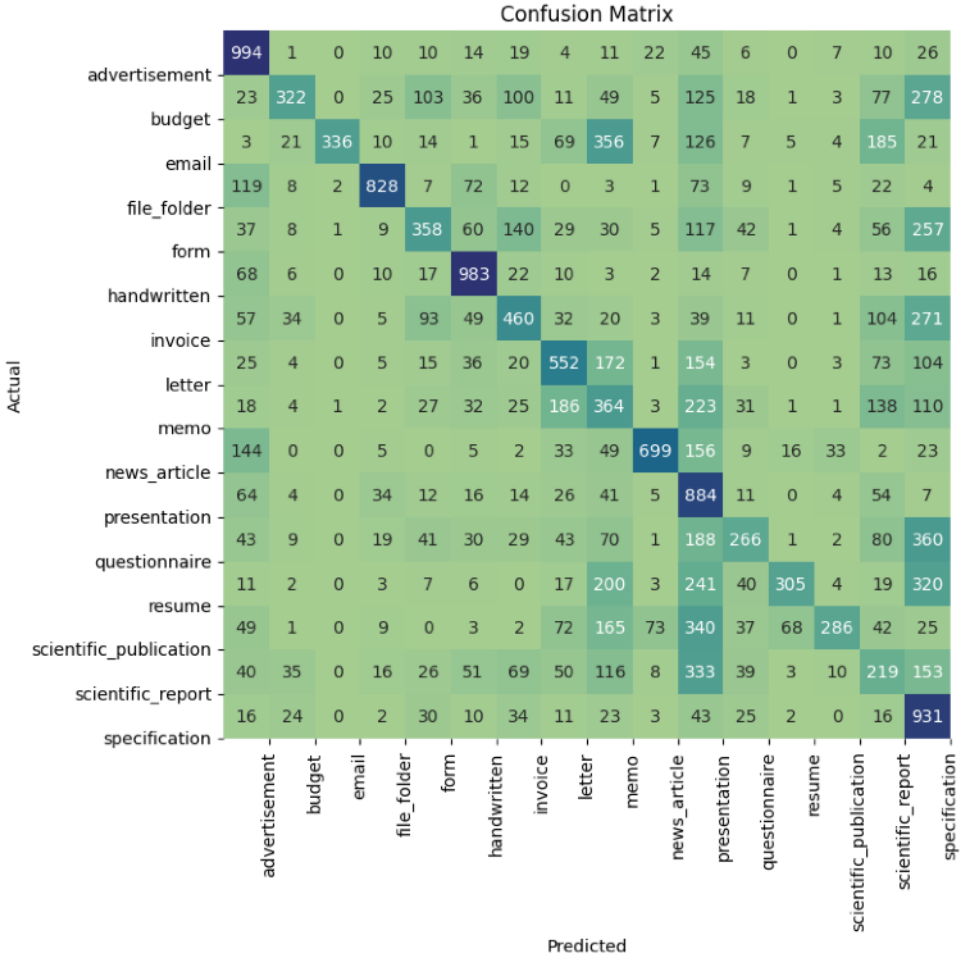


Fig. 2 Basic CNN

4.2) AlexNet

Classification Report

	precision	recall	f1-score	support
advertisement	0.5809	0.8431	0.6879	1179
budget	0.6667	0.2738	0.3882	1176
email	0.9882	0.2847	0.4421	1180
file_folder	0.8347	0.7101	0.7674	1166
form	0.4711	0.3102	0.3741	1154
handwritten	0.7001	0.8387	0.7632	1172
invoice	0.4777	0.3902	0.4295	1179
letter	0.4821	0.4730	0.4775	1167
memo	0.2177	0.3122	0.2565	1166
news_article	0.8312	0.5944	0.6931	1176
presentation	0.2851	0.7517	0.4134	1176
questionnaire	0.4742	0.2250	0.3052	1182
resume	0.7550	0.2589	0.3856	1178
scientific_publication	0.7772	0.2440	0.3714	1172
scientific_report	0.1973	0.1875	0.1923	1168
specification	0.3204	0.7957	0.4568	1170
accuracy			0.4684	18761
macro avg	0.5662	0.4683	0.4628	18761
weighted avg	0.5666	0.4684	0.4628	18761



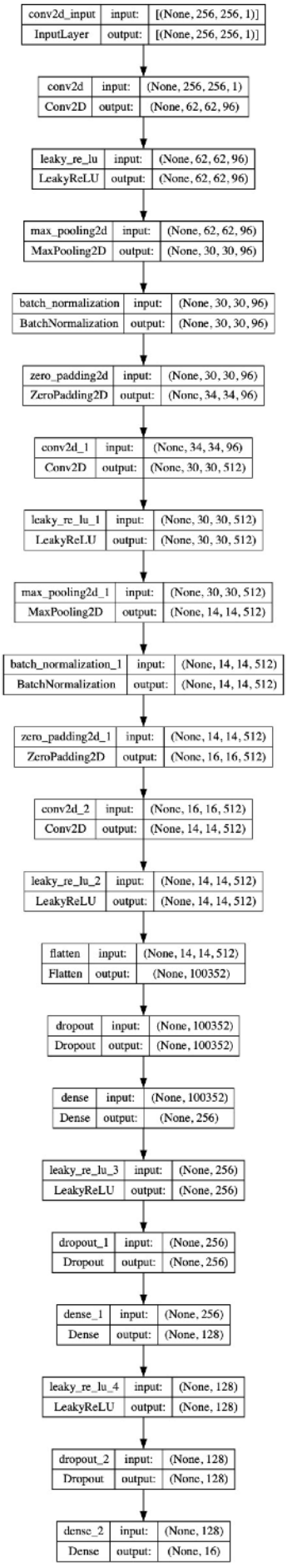
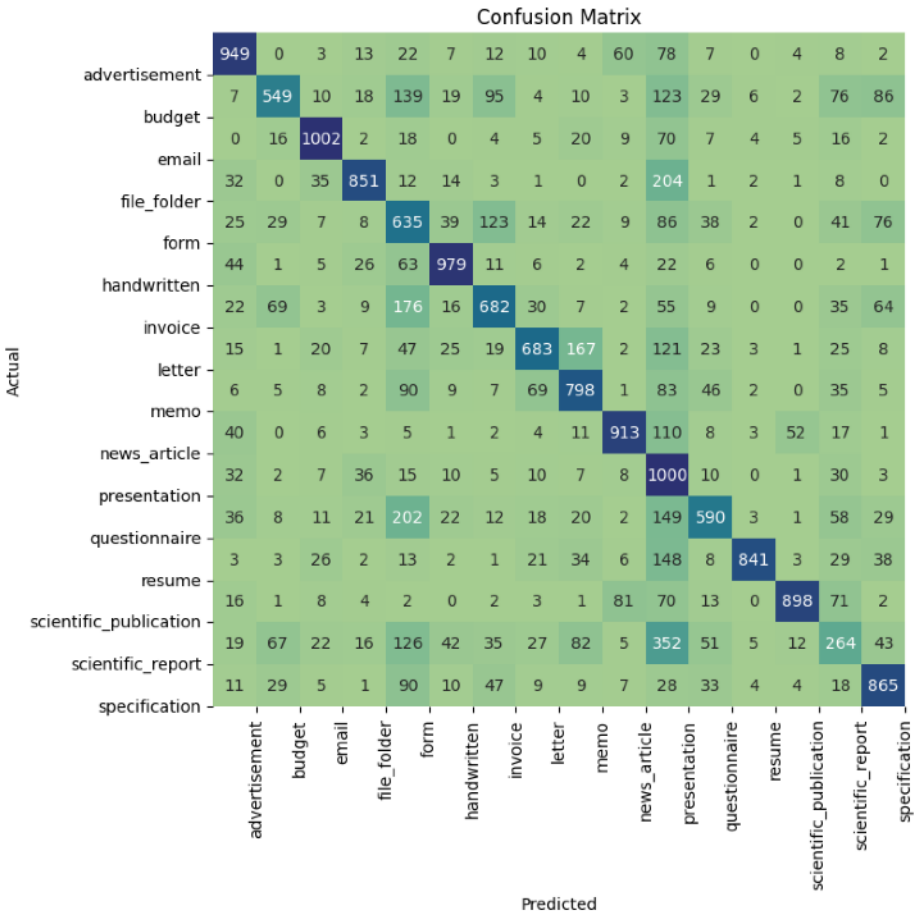


Fig. 3 AlexNet

4.3) VGG-19

Classification Report

	precision	recall	f1-score	support
advertisement	0.7550	0.8049	0.7791	1179
budget	0.7038	0.4668	0.5613	1176
email	0.8506	0.8492	0.8499	1180
file_folder	0.8351	0.7298	0.7789	1166
form	0.3837	0.5503	0.4521	1154
handwritten	0.8192	0.8353	0.8272	1172
invoice	0.6434	0.5785	0.6092	1179
letter	0.7473	0.5853	0.6564	1167
memo	0.6683	0.6844	0.6763	1166
news_article	0.8196	0.7764	0.7974	1176
presentation	0.3705	0.8503	0.5161	1176
questionnaire	0.6712	0.4992	0.5725	1182
resume	0.9611	0.7139	0.8193	1178
scientific_publication	0.9126	0.7662	0.8330	1172
scientific_report	0.3602	0.2260	0.2777	1168
specification	0.7061	0.7393	0.7223	1170
accuracy			0.6662	18761
macro avg	0.7005	0.6660	0.6706	18761
weighted avg	0.7009	0.6662	0.6709	18761



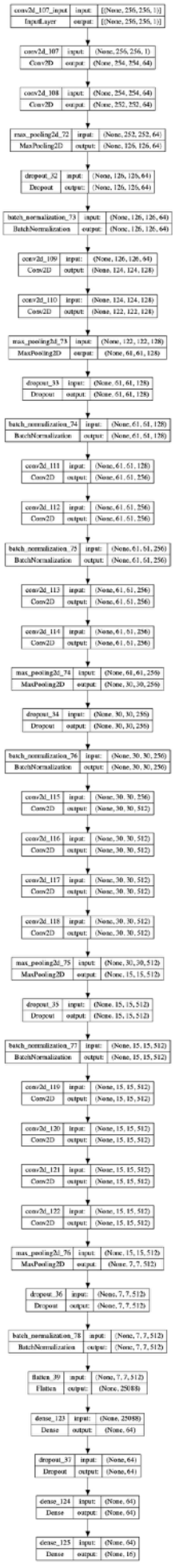


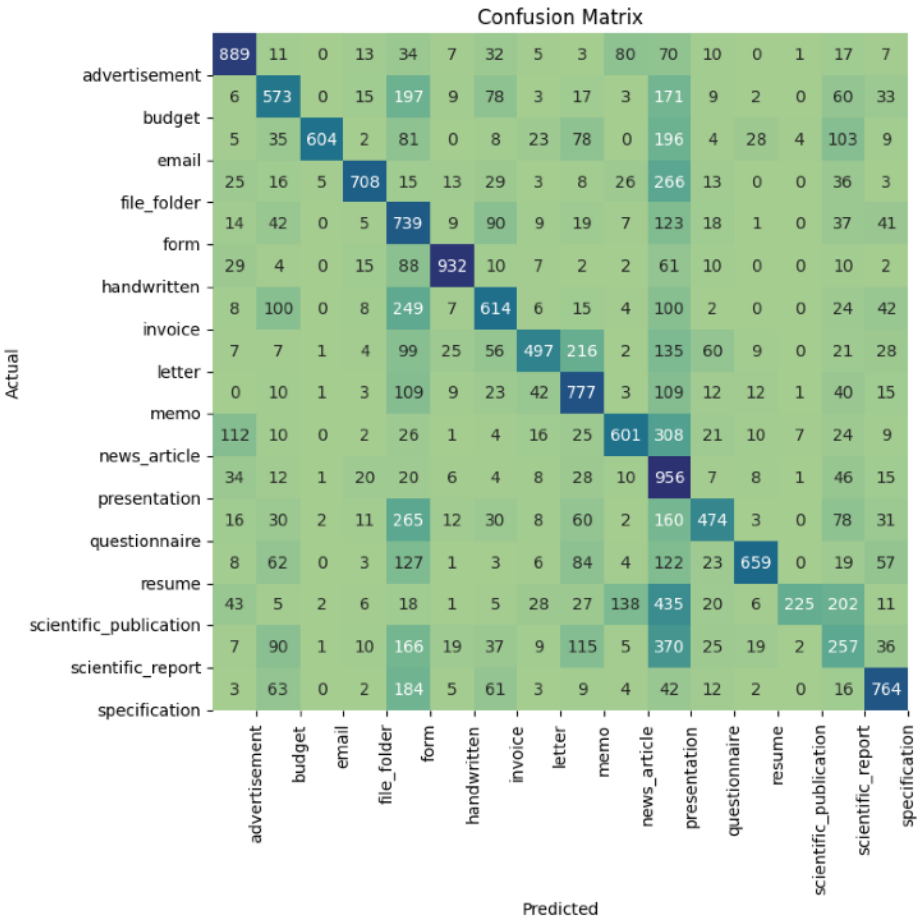
Fig. 4 VGG-19

4.4) ResNet

Note: Resnet Architecture has been cropped to fit here (it does not include the complete layers)

Classification Report

	precision	recall	f1-score	support
advertisement	0.7371	0.7540	0.7455	1179
budget	0.5355	0.4872	0.5102	1176
email	0.9789	0.5119	0.6722	1180
file_folder	0.8561	0.6072	0.7105	1166
form	0.3058	0.6404	0.4139	1154
handwritten	0.8826	0.7952	0.8366	1172
invoice	0.5664	0.5208	0.5426	1179
letter	0.7385	0.4259	0.5402	1167
memo	0.5239	0.6664	0.5866	1166
news_article	0.6745	0.5111	0.5815	1176
presentation	0.2638	0.8129	0.3983	1176
questionnaire	0.6583	0.4010	0.4984	1182
resume	0.8682	0.5594	0.6804	1178
scientific_publication	0.9336	0.1920	0.3185	1172
scientific_report	0.2596	0.2200	0.2382	1168
specification	0.6927	0.6530	0.6722	1170
accuracy			0.5474	18761
macro avg	0.6547	0.5474	0.5591	18761
weighted avg	0.6552	0.5474	0.5594	18761



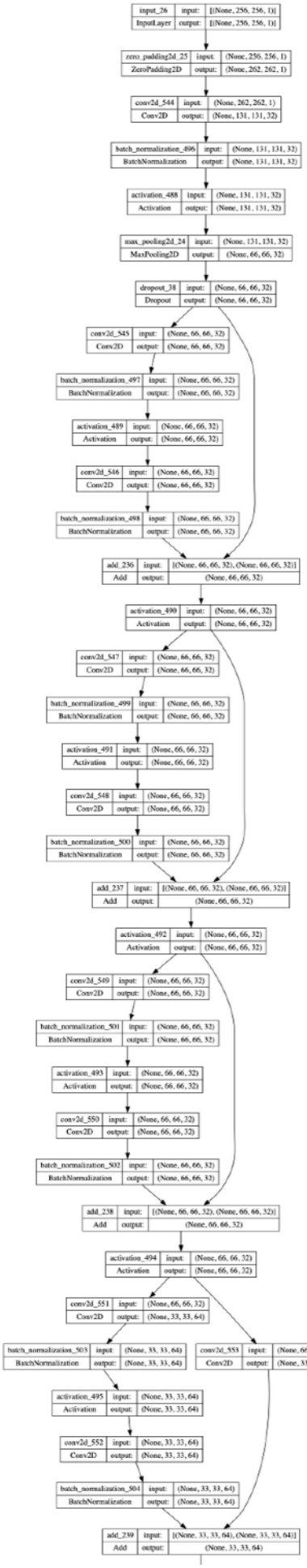


Fig. 5 ResNet

4.5) Model Comparison & Conclusion

	Accuracy	Precision	Recall	F1-Score	Parameters	α
Basic CNN	60.89	64.22	60.88	62.505	21,60,528	28.9
Alex Net	46.84	56.62	46.83	51.27	2,93,27,376	1.7
VGG-19	66.62	70.05	66.60	68.281	2,16,37,584	3.2
ResNet	54.74	65.47	54.74	59.626	70,36,560	8.5

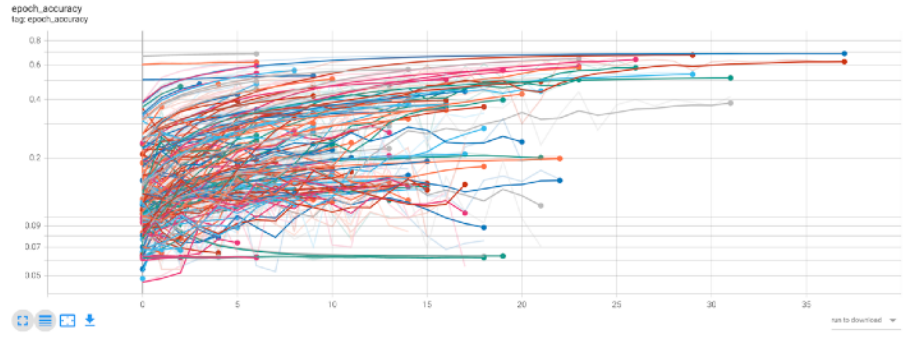
Note: All values are in percentage

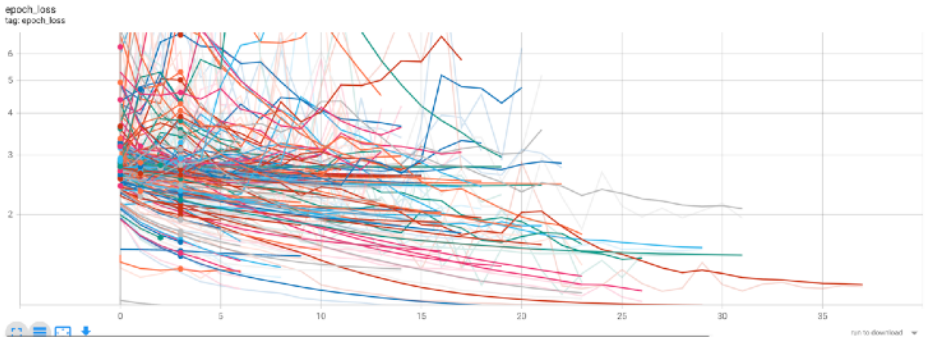
Looking at the data it is clear that the **VGG-19 is performing the best** out of all the models. No transfer learning was used to train all these models. These models have been trained from scratch on a personal laptop. Configuration of the laptop (Apple Macbook Pro M1) used was

- *Chip* - Apple M1 Pro (GPU + CPU embedded)
- *Memory* - 16 GB
- *OS* - MacOS Ventura 13.3.1
- Tensorflow version - 2.12.0 (tensor flow-metal)
- Python Version - 3.8
- IDE - IntelliJ

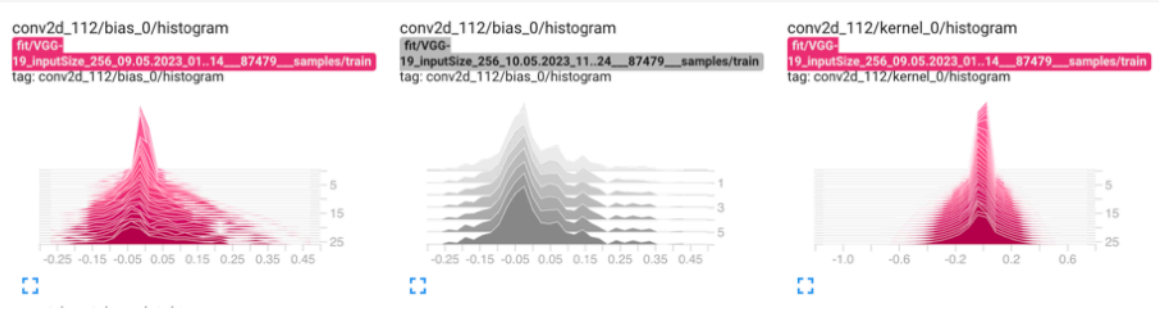
Important Note Regarding the Dataset *- It is worth mentioning that we started by creating a smaller dataset which contained 1000 samples in each class, and we trained and tuned a lot of models on this dataset. However, the results weren't promising enough to be included in this report. So it was decided to increase the data from 16000 images to approximately 1,00,000 images, which is again a subset of the original RVL-CDIP dataset but led to significant overall improvement which has been reported. These models take huge time to train, sometimes 2-3 days, like VGG took around 56 hours to train. So, these all models are all trained, validated and tested on a complete set of 1,00,000 images.*

*Here's what we mean by "**trained a lot of models**"*





There are a lot more things we could comment on like the histograms of the weights that could be seen using the tensor board. This allows us to visualise the weights distribution changing over time but is not a part of this report. Here's an example -

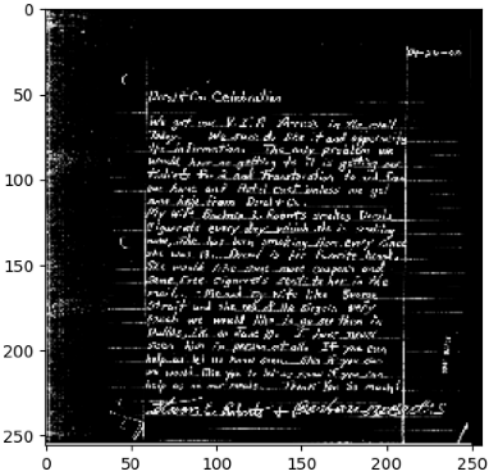


Also, an **alpha** parameter is included in the comparison table above which is nothing but the ratio of the F1-Score and number of trainable parameters and scale by 10^6. **α** is an important parameter to consider, as if we just look at the F1-Score, it makes sense to choose VGG-19 model, but the training time is huge as compared to the Basic CNN model, that consisted of only a fraction of the parameters in the VGG model and also was able give decent classification results. So, the practical option is to go with the Basic CNN Model, with α=28.9, which isn't as resource demand than the best option available i.e VGG-19.

5. Appendix

Hidden Layer Activation extraction for VGG-19

Sample Image

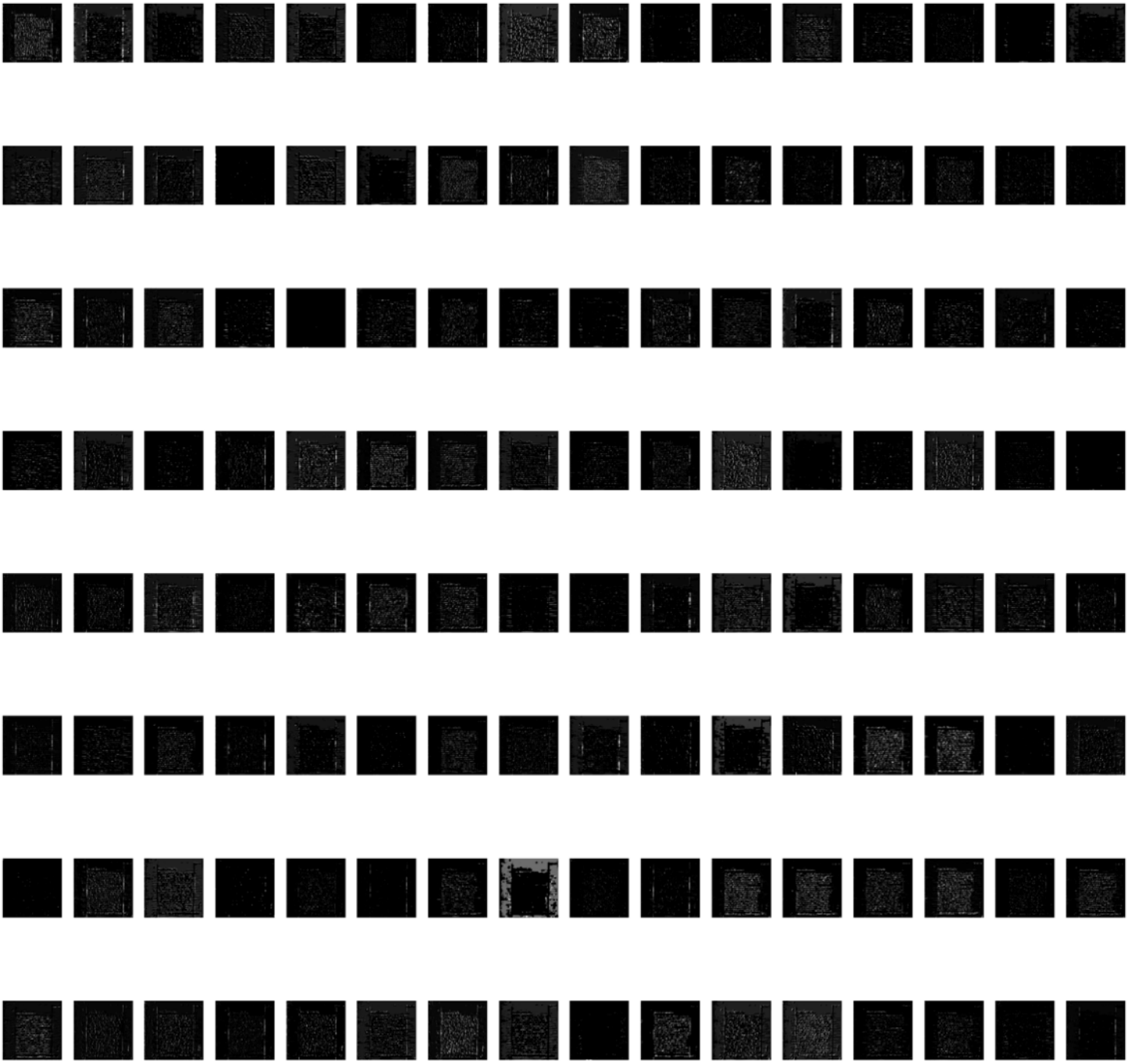


To keep this report short, only activation images of VGG-19 (first few layers) are shown here. It is not possible to put activations of all the trained models in this report. So, we'll try to summarise our learning here.

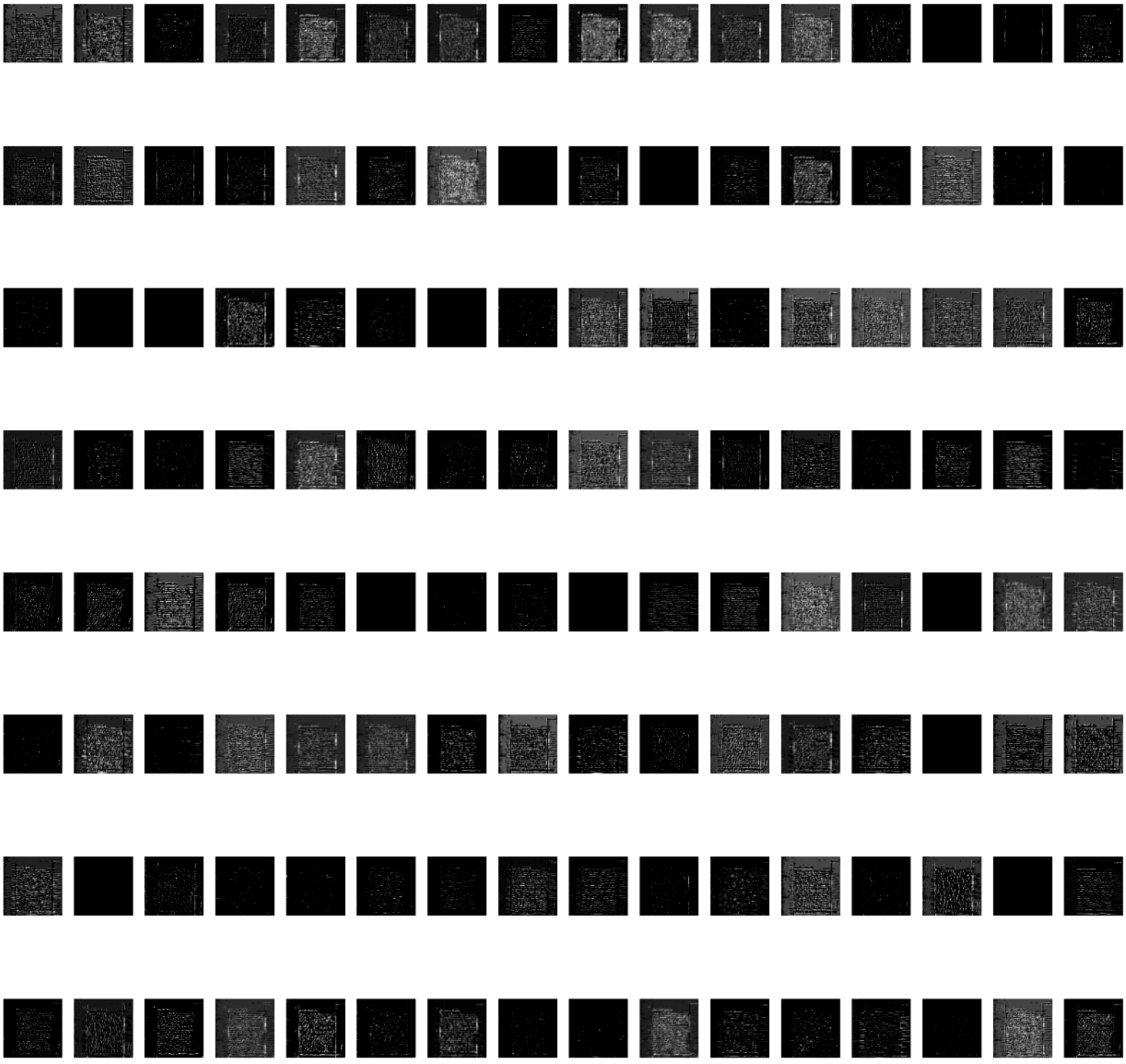
- Most of the deep architectures tend to compress the documents to such a level that they don't even retain their basic structure, so models trained on MNIST Fashion dataset may have a good starting point but last few layers need to be adjusted according to the problem in hand.
- Alexnet with Relu as activation function, seem to have many black images (no activations), couldn't figure of the reason of this, but using leaky rely did help a bit.
- Since we are dealing with documents and not objects, we need to ensure that the final convolution layer shouldn't compress the image beyond recognition (one example would be shown at the end).
- More filters in the initial layer and less number of layers actually improve model performance

1st Layer - VGG

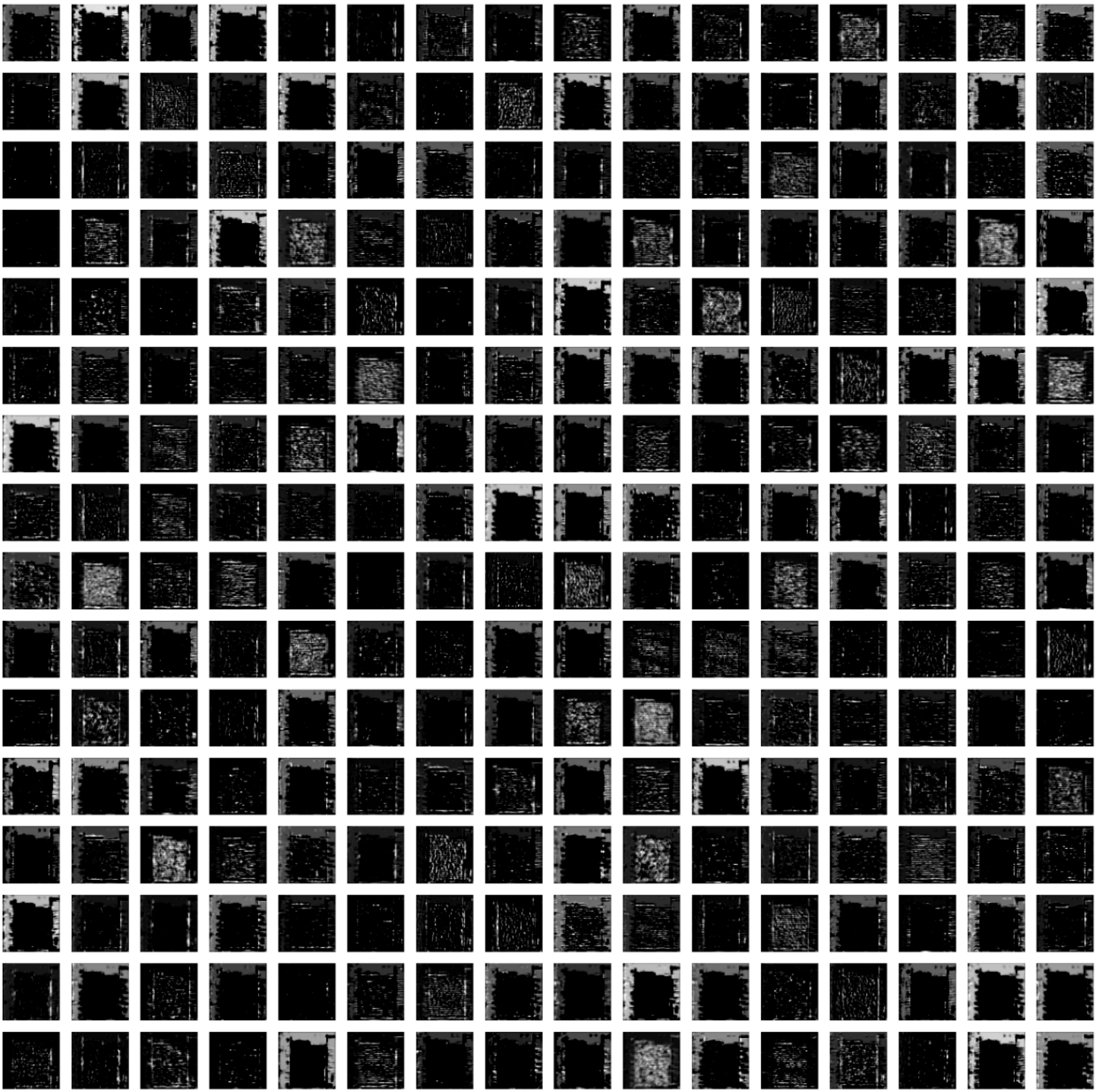
2nd Layer - VGG



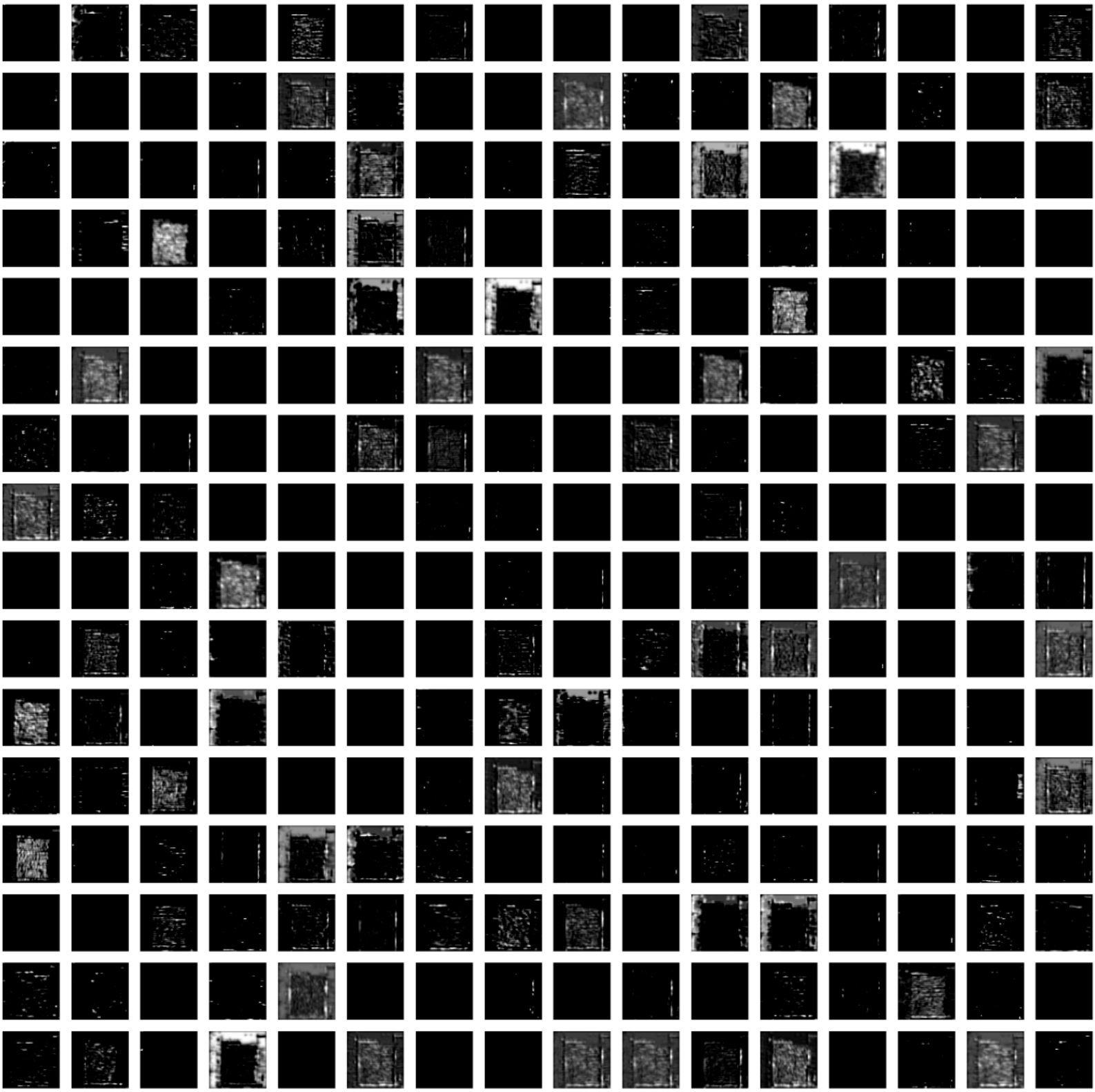
3rd Layer - VGG



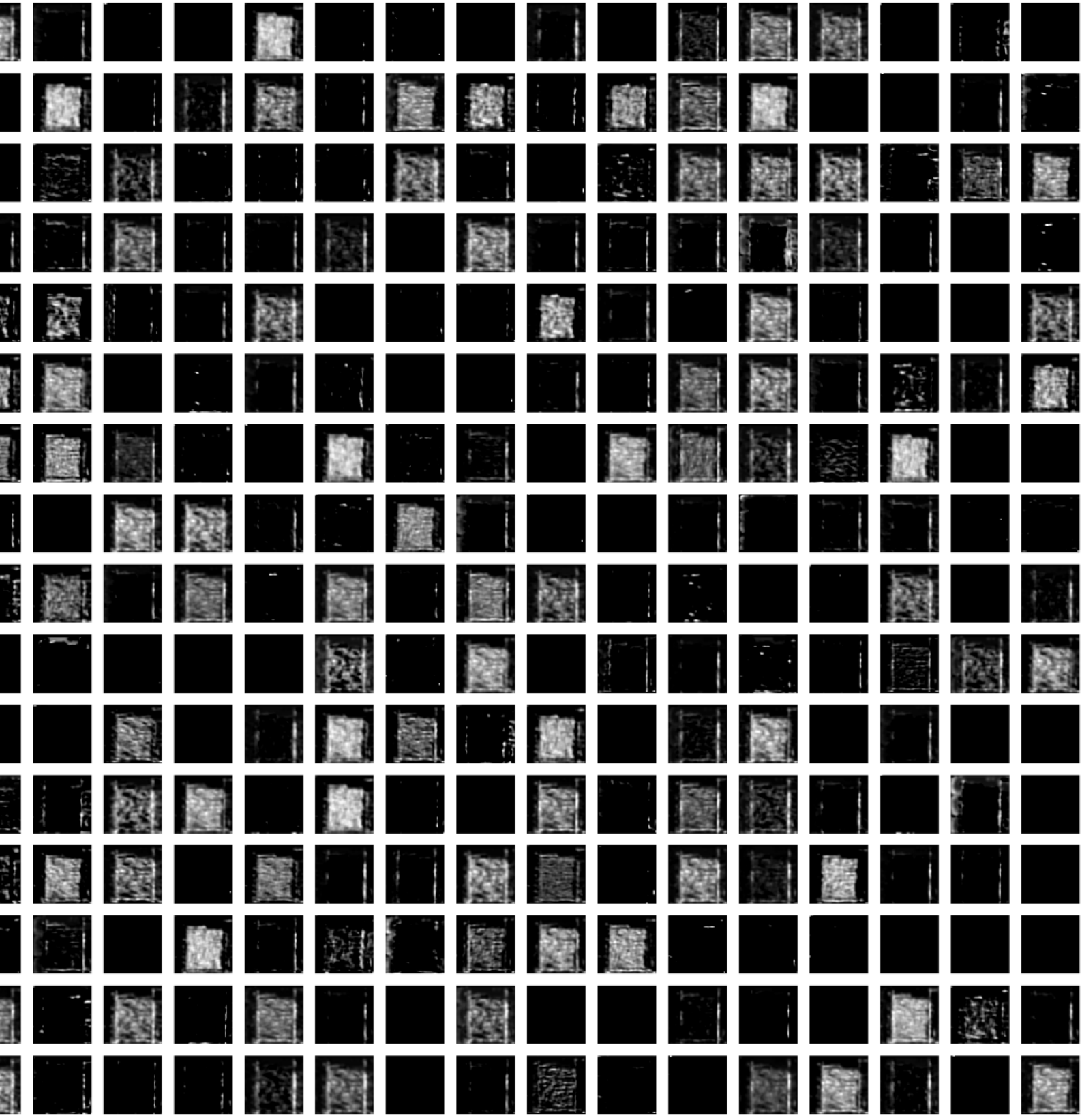
4th Layer - VGG



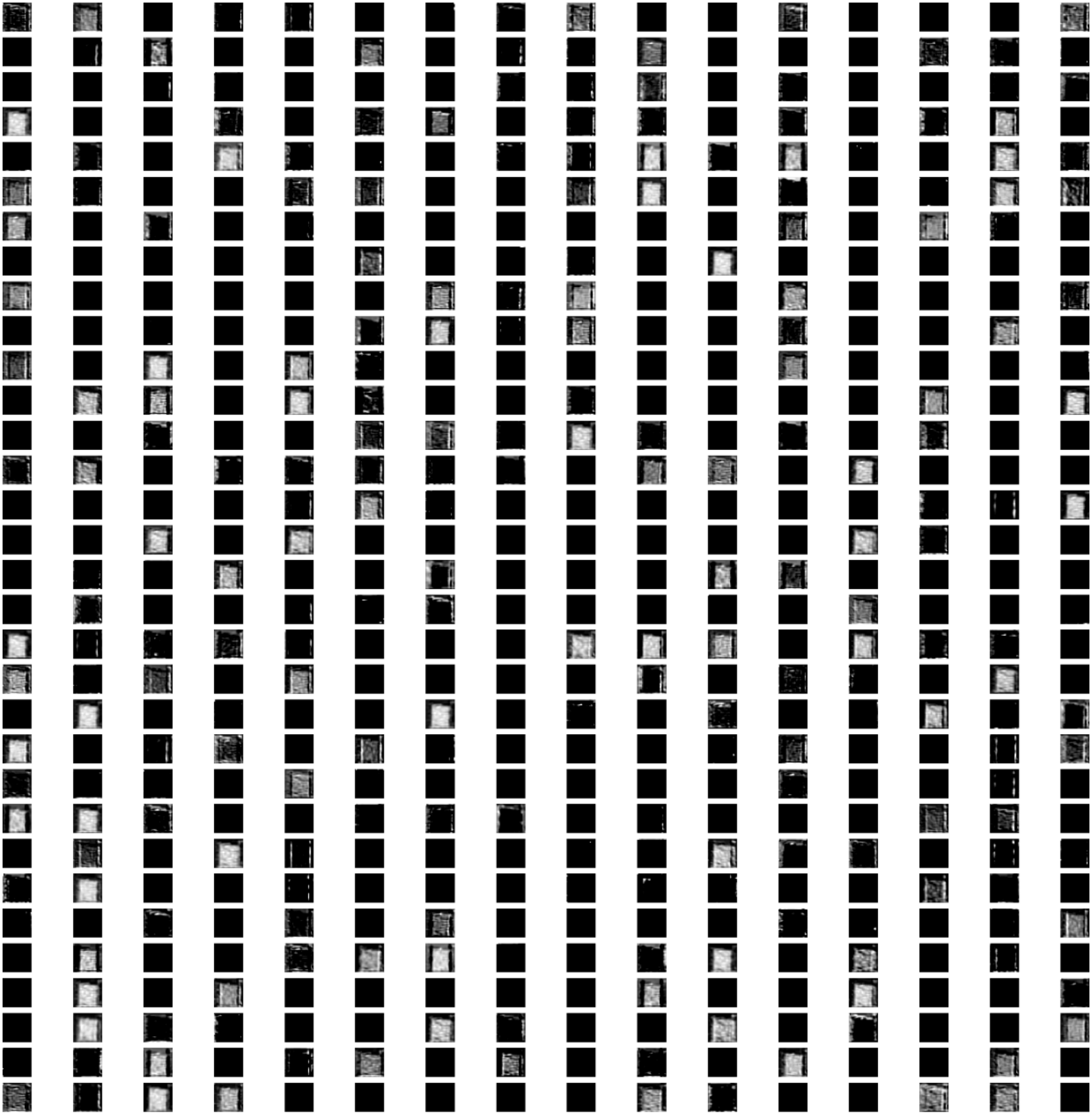
5th Layer - VGG



6th Layer - VGG



7th Layer - VGG



10th Layer - VGG

More work into this project is needed to infer all these activations for each of the models.