Utrecht University
Utrecht, Summer 2020

Classifying digital documents reliably using machine learning

Hristo S. Hristov

Master thesis, Computing science

UTRECHT UNIVERSITY
ICA-6307973

Acknowledgements

The work on this thesis has been done in collaboration with Radial SG and Utrecht University. I'm grateful for the financial support for the development of this thesis provided by Radial SG. This work would also not be possible without the support of my family, friends and the help of my supervisors.

Utrecht University

Utrecht, July 2020

Student, Hristo S. Hristov	Direct supervisor, Mark van 't Zet
First supervisor, dr. Ad Feelders	Second supervisor, prof. dr. A.P.J.M. Siebes

Contents

List of Figures

List of Tables

1 Introduction

1.1 Abstract

Digital documents are the de facto container to transfer information in our age and due to the growing prevalence of large amounts of documents, organizations may struggle to organise and categorize these documents into their respective categories. In this work we show how a system that uses real-life documents can use scaling of the neural network architecture EfficientNet and its input images and we compare its performance with the widely used VGG-16 architecture. We explore how we can augment the input data through RandAugment, a data augmentation strategy which has only 2 parameters which allows the user to search through the search space of possible augmentations faster. We show that transforming the input images can cause a classifier which is not trained on transformed images to perform poorly. We adapt a multi modal fusion method that has not, to this point, been applied to document image classification - tensor fusion. We propose suitable parameters for this approach and compare it to the more standard early and late fusion approaches by training our models on a balanced subset of the widely used Tobacco-3482 dataset. We then apply this approach to our own dataset and draw conclusions on which approach is the most appropriate. Our results show that tensor fusion can be more successful than late fusion but stops short of the performance of the simpler to implement early fusion. Additionally, tensor fusion requires more hyper parameters to be considered which makes the implementation of this approach more difficult. By comparing the results of humans on the ImageNet dataset, we argue that the top 3 predictions of our approach can provide reliable category recommendations.

1.2 Preface

Radial SG is a software product company based in Woerden, the Netherlands. Since its start in 2010, it has actively answered the data management demands of industry. The Viewport solution was created to solve real world problems involving information understanding and retrieval by using the latest techniques in artificial intelligence and machine learning. It understands various types of documents, automatically finds relations between documents and organises information in a convenient way. In Viewport, categories of document images are encountered in vast amounts whose correct classification is crucial to providing value to the users. This is what prompted us to research what the state of the art approaches to document image classification can provide. The goal is to create a model that performs as well as possible on unseen real-life data.

1.3 Problem definition

This work focuses on the problem of document image classification. Document image classification is a subset of the image classification problem in which the task is to assign a single-page document image to one of a set of predefined document classes [Chen and Blostein (2007)]. There is no precise definition of a document [Baltrusaitis et al. (2017)]. In this work a document refers to a single-page document image. Example documents are presented in figure 4.1. [Nagy (2000)] defines 3 distinct types of document image classification tasks and a taxonomy of commonly used documents.

The 3 distinct types of document image classification problem types are illustrated in figure 1.1. The *document space* is the range of samples of input documents which a classifier is expected to handle [Chen and Blostein (2007)]. Gray circles represent the document space while colored circles represent document classes.

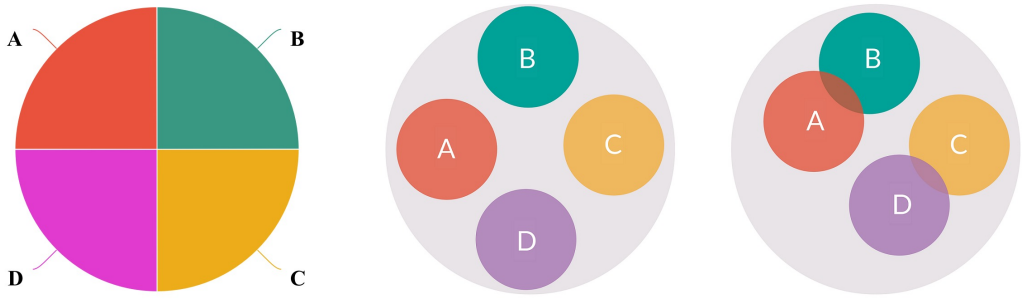


Figure 1.1: The three document image classification problem types

1. Full coverage: The document space is fully covered by the 4 distinct classes.
2. Partial coverage: The document space is partially covered by the 4 distinct classes.
3. Partial coverage with intersecting categories: The document space is partially covered and some documents may have multiple labels.

In this thesis we're interested in solving the partial coverage version of this problem.

1.4 Problem relevance

The document image classification problem is relevant in the context of automatic distribution or archiving of documents, creation of digital libraries, document image retrieval and higher level document analysis [Chen and Blostein (2007)]. A use for the

document image classification problem has been identified in several industries including passport verification at airports, identifying eye conditions and pneumonia [Kermany et al. (2018)], video analysis [Snoek et al. (2005)], classifying documents in banking [Engin et al. (2019)] and classifying medieval handwriting [Pondenkandath et al. (2017)]. Due to the abundance of such documents in high resolution and their common use, they represent a good target for machine learning.

1.5 Relevance of this research

With the increasingly large amount of data available online there is more and more information requiring human attention. Therefore, if we can reduce the amount of time human attention is not used optimally we can save time. Libraries, archives and other institutions dealing with large amounts of documents can categorize incoming documents with or without human intervention. The following stakeholders for the document image classification problem may have interest in this research:

1. The system users have an interest in receiving answers in real-time from document retrieval systems. In many real-life cases this can only happen if the system doesn't rely on humans to perform document image classification.

2. Safety critical organisations - a document image classification system for a nuclear station may require equal or better than human accuracy in the classification of documents.

3. Non-safety critical organisations have an interest in using automatic classification to reduce costs of human labour and lower the proneness to error.

2 Background

2.1 Artificial neural networks

This research uses artificial neural networks (ANN) for all experiments. Therefore, we first summarize the necessary background knowledge in order to understand the content of this work.

2.1.1 What is an artificial neural network?

A neural network is a computational model invented in the last century which has grown to be increasingly used in the scientific community due to the way it can represent arbitrary functions given sufficient data and depth [Leshno et al. (1993)]. Typical artificial neural networks draw inspiration from the human brain. More specifically, human neurons, their behavior and the synapses connecting them. Such a network is often represented by a series of layers which are combined to create the whole network. There are different kinds of layers which can have different effects on the input. For example, a fully connected layer (often referred to as *dense* layer) can be seen in figure 2.1. We refer to the overall composition of such a network as an architecture.

2.1.2 Neurons and weights

Each neuron takes in the values of input neurons and multiples it with a real value weight $w^l_{j,k}$ of the connection between neurons j in layer l and k in layer $l-1$. a^{l-1}_k is the activation of neuron k in layer $l-1$. Each neuron also has a bias value b^l_j which is added to the sum of those values.

The output of the j-th neuron in layer l is:

$$a^l_j = \phi\left(b^l_j + \sum_{k=1}^{M} w^l_{jk} a^{l-1}_k\right) \tag{2.1}$$

where M is the number of neurons in layer $l-1$ and ϕ is an activation function.

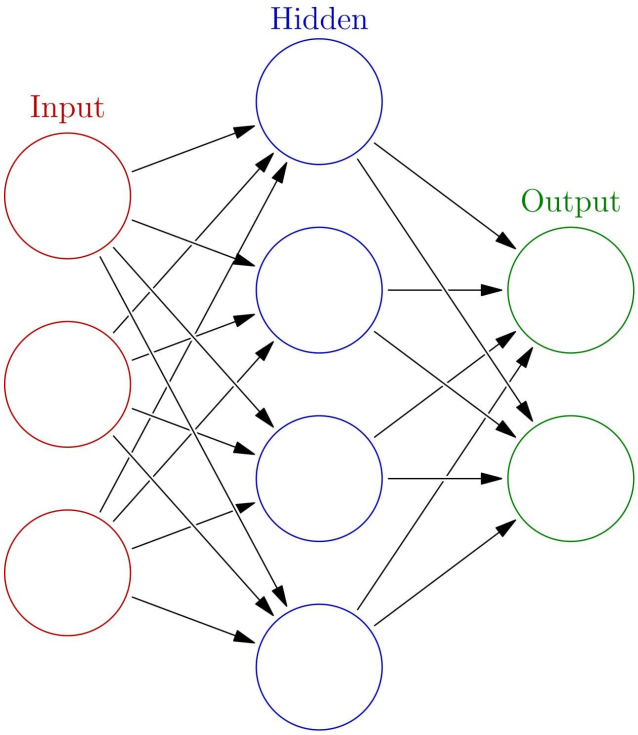


Figure 2.1: An example of a neural network containing 1 input (dense) fully connected layer, a fully connected hidden layer and a fully connected output layer. Source By Glosser.ca - Own work, Derivative of File:Artificial neural network.svg, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=24913461

2.1.3 Activation functions

For this work we use the three activation functions listed below.

$$1.\ \text{ReLU} - relu(v) = \begin{cases} v & v > 0 \\ 0 & v \leq 0 \end{cases}$$

$$2.\ \text{Sigmoid} - h_\theta(v) = \frac{1}{1+e^{-\theta^T v}}$$

$$3.\ \text{Softmax} - s(\vec{x}, i) = \frac{\exp(\vec{x}_i)}{\sum_{j=1}^{K} \exp(\vec{x}_j)}$$

The softmax function can be used to assign probabilities to each class in the final layer of a neural network. This is useful when solving multi-class classification problems. K is the number of classes and $\vec{x}$ is a vector which contains all the values (as specified in equation 2.1) from the final layer.

2.1.4 How to train an artificial neural network?

An artificial neural network (ANN) or often in related literature known as neural network is trained through the back propagation algorithm which through the use of a loss function and gradients of that function is being continuously updated with the goal to decrease the value of the loss function. Decreasing the value of the loss function aims to achieve improvements in overall performance of the model. In order to continuously check the value of the loss function and possibly other metrics, often a small (10%) subset of the dataset is used. This data set is called the validation set. After the training is done we test the resulting model on a test set.

2.1.5 Classification problems and loss functions

A classification problem is a problem in which instances drawn from some distribution are given and the goal is to predict to which of a set of given categories each of those instances belongs to. In classification problems two loss functions are commonly used - the *categorical cross entropy* loss function and the *Kullback-Leibler divergence* function. In this work we will focus on the categorical cross entropy loss function as defined in equation 2.2. In this equation $y_{n,k}$ is 0 or 1 and indicates whether class label k is the correct label and $\hat{y}_{n,k}$ denotes the predicted probability that sample n belongs to class k. N is the number of samples and K is the number of classes. We chose this function as it makes comparison with other research in the area easier. It was used in [Afzal et al. (2017), Engin et al. (2019), Audebert et al. (2020)].

$$L = -\frac{1}{N}\sum_{n}^{N}\sum_{k}^{K} y_{n,k} \log \hat{y}_{n,k} \tag{2.2}$$

2.1.6 Transfer learning

Several studies [Tan et al. (2018), Yosinski et al. (2014)] show that we can initialize the weights of a neural network by copying the weights of an already trained network which has the same architecture. This tends to reduce the computational time required to train a neural network and can result in improvements of accuracy. This approach has been widely adopted in various industries such as bioinformatics and robotics. A common

example of transfer learning is taking the weights from a model trained on the popular ImageNet dataset and training the model with those weights as a starting point. This can be done with other data sets. In the area of research for the problem of document image classification, a common weight transfer is from a model trained on the RVL-CDIP dataset (see section 4.1.1 for more details).

2.1.7 Training in batches

Being able to efficiently estimate the gradient of the loss function is one of the main problems that neural networks face. Computing the gradients on the whole dataset may be far too expensive for the GPU even with modern video cards such as the NVidia K80. It is more common to use a small (compared to the dataset) batch size to compute an approximation of the gradient. A typical value is 32 but depending on the size of the input that can be either reduced or increased.

By setting the batch size, we can adjust how much of the memory of the GPU is used during training. At the expense of training time, we may lower the batch size, which also has been shown to reduce over fitting [Keskar et al. (2017)].

2.1.8 The Adam optimizer

Training a neural network is done by an optimizer. The task of the optimizer is to specify the way weights in the neural network are updated each iteration. In recent literature the Adam optimizer [Kingma and Ba (2014)] has proven to perform well and has become commonly used in training of neural networks. It includes an adaptive learning rate which means that the learning rate is automatically adjusted during training.

2.1.9 Regularization and over fitting

A notable problem that occurs during training of ANNs is that the model learns to predict the items in the training set far too well and loses the ability to generalize to new examples. This results in a deterioration of the performance on the test set. To avoid over fitting dropout layers are used: a dropout layer randomly sets input units to 0 with a user specified rate at each step during training time. [Srivastava et al. (2014)] proposed that a drop out rate of 0.5 is suitable for most networks.

2.1.10 Early stopping and lowering learning rate on plateaus

Training deep neural networks, or networks which have a large number of consecutive layers, costs a significant amount of GPU time. Furthermore, it requires GPUs with a large amount of memory in order to estimate the gradients well enough. Therefore, we want to save as much computing power as possible and we would like to stop training once we no longer are learning information that generalizes to unseen examples. We monitor the value of a chosen metric of the validation set and if no improvement is seen for the value for a number of epochs (often called patience), then the learning rate is reduced. We also use a procedure to check if we're in a plateau, if yes, then we lower the learning rate in order to attempt to find a way to continue learning. Models often benefit from reducing the learning rate once learning stagnates.

2.1.11 Convolutional neural networks

Convolutional neural networks (CNNs) are a special type of ANN which have become increasingly popular in scientific literature due to their ability to use the spatial similarity of features in order to learn to predict unseen examples. For example, in an image, pixels which are closer to each other have more in common than those which are further away. Convolutional neural networks exploit that to reduce the search space and, consequently, improve the performance of models. A convolutional neural network applies filter regions that extract features. Pooling layers can be used to reduce the dimensionality of the network. For example, average pooling takes the average value in a filter region. A typical example of a convolutional neural network architecture is shown in figure 2.2. Convolutional neural networks can be used as feature extractors [Hertel et al. (2015)]. This is particularly important as we require a way to encode a document into a feature vector, also known as embedding.

2.1.12 Model evaluation

In this research we use two techniques to measure and compare the performance of models depending on the data set used. One of the data sets used in this work, the T3482 data set, contains only 3482 samples. In order to avoid unrepresentative results we follow the dataset balancing technique used in [Afzal et al. (2017)]. This is described later in chapter

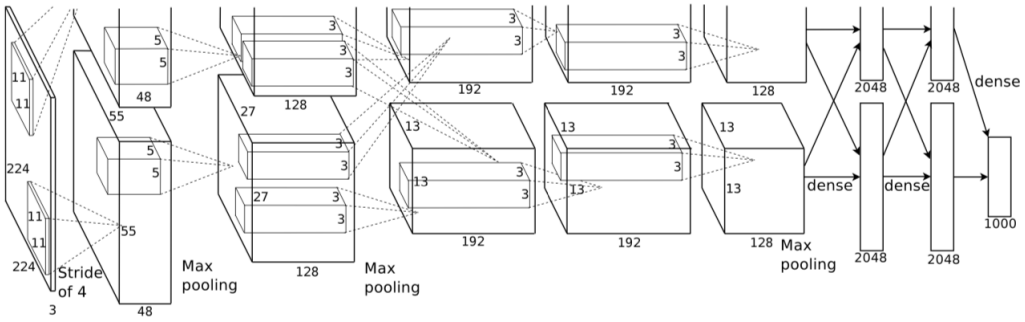


Figure 2.2: The AlexNet architecture, image by Krizhevsky et al.

5.2.2. In this case the main performance metric used is accuracy. For the INTERNAL dataset we track 6 different metrics. The F1 score is the leading metric used for the INTERNAL data set.

1. Loss function - for all experiments we track the value of the loss function categorical cross entropy.

2. Top 1 accuracy - this corresponds to the standard definition of accuracy. We take the max value of the softmax layer outputs. This corresponds to the label for which the classifier is the most confident it corresponds to the input features.

3. Top 3 accuracy - in this metric the output probabilities of the model for each class are sorted in descending order and the top k items are picked. If the true label is part of those k items the output is true for the sample. This is important for us because if the Top 1 accuracy isn't high enough then the top - 3 answers can be used instead.

4. Average precision - a metric for multi-class classification of how many selected items are relevant.

5. Average recall - a metric for multi-class classification of how many relevant items are selected.

6. F1 score - a metric that computes the harmonic average of the precision and recall.

2.2 Multi modal fusion

2.2.1 What is a modality?

A modality is a source of information, a particular mode in which something exists or is experienced or expressed. In the representation learning area of research, the word "modality" refers to a particular way or mechanism of encoding information [Guo et al. (2019)]. For example, in the case of document image classification, there are two modalities - the text written on the document and the image of the document.

2.2.2 Embedding

Embedding in the context of this work refers to mapping samples of a given modality into a subspace. This allows us to perform various operations on the mapped values and understand how objects relate to each other. If we are given a text modality, a subspace constructed from a set of sample text can allow us to see which words are similar to each other with respect to the way the embedding is constructed.

2.2.3 Multi modal fusion

Multi modal fusion is a field whose focus is on learning how to combine different modalities in order to create better models. A typical example of a situation where combining multiple modalities may be beneficial to a learner is when a classifier is to classify whether a movie is good or bad. When somebody says "*This movie is sick!*" it can mean different things depending on the expression of the person who says that. A negative expression might mean that the movie is indeed horrible, while a positive expression indicates approval of the movie. In this case there are two modalities - visual (the expression of the person) and their speech. Multi modal fusion in essence attempts to solve a common problem in the human brain - how to combine information from different sources into a good representation of reality. In the context of this work, we use two modalities, text and image. The goal is to fuse the modalities in to a common representation by mapping them into a common subspace. This can be seen in figure 2.3.

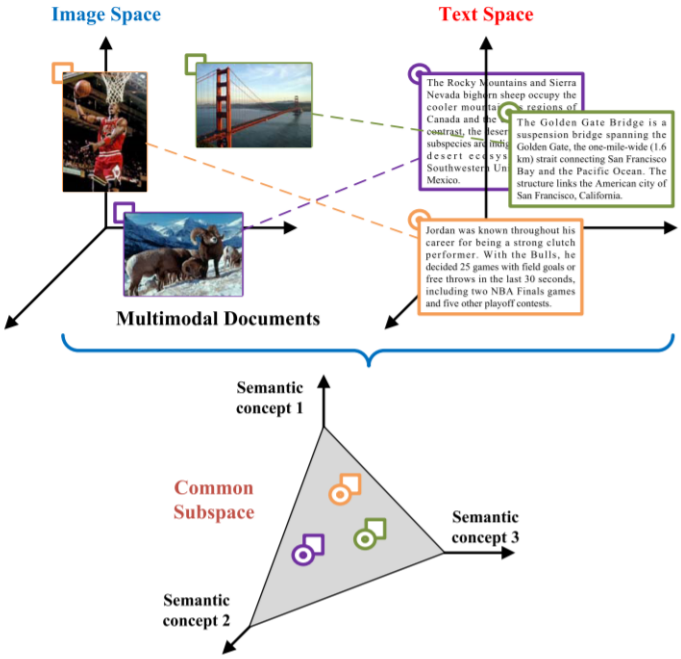


Figure 2.3: Multi modal fusion tries to find a common subspace between the various semantic concepts. Source - [Guo et al. (2019)]

2.2.4 Early fusion

Early fusion is a method to combine features extracted from a set of samples from one modality with other modalities in order to create a common representation. In this work, we first obtain embeddings of each modality. For the image modality, that is done by the **Image** model while for the text modality we use fastText [Bojanowski et al. (2016)] to get word embeddings. These word embeddings are then transformed to a document embedding by mapping the word embeddings of a document into a single vector. The simplest way to construct such a document embedding is to perform some arithmetic operation on the word embeddings of the document in order to create a document embedding. Two of the most commonly used operations are summation and averaging. In this work we use the averaging operation [Kenter et al. (2016a)]. This same strategy was used in [Audebert et al. (2020), Engin et al. (2019)]. After obtaining these two representations they are merged into a common representation. An often used approach to merging the representations is to concatenate them.

2.2.5 Late fusion

Late fusion is a method to combine the output probabilities for the classes of the individual classifiers for each modality. Each classifier learns on its own and the predictions are combined with a strategy. One may use a heuristic based strategy (for example, pick the classifier which is the most confident) but in this work we will focus on model based strategies. A useful review of possible heuristic strategies can be found in the experiments performed in [Åberg (2018)].

2.2.6 Tensor fusion

We include an approach that has not been applied to document image classification to this point, to our best knowledge. It is inspired by the work done in [Zadeh et al. (2017)] and later in [Liu et al. (2018)]. In this approach tensors for each modality are obtained and combined to generate a multi modal representation. The difference between the tensor fusion approach and the concatenation approach in early fusion is that tensor fusion also accounts for the the uni modal representation with addition to the bi modal representation of the data. This should allow us to model the intra modal (within a single modality) interactions more effectively. In comparison, in early fusion the concatenation layer acts as a compression layer for both the unimodal and bimodal features while in tensor fusion the unimodal and bimodal features are modelled more explicitly and therefore, should be more expressive.

This operation is illustrated in equation 6.1 where z^i and z^t are the tensors from the image and text modalities respectively and $\bigotimes$ represents the outer product operation.

$$z^m = \begin{bmatrix} z^i \\ 1 \end{bmatrix} \bigotimes \begin{bmatrix} z^t \\ 1 \end{bmatrix} \tag{2.3}$$

This operation, however, proves to be computationally expensive when the number of modalities becomes higher. In order to address this issue [Liu et al. (2018)] tries to create a low rank version of the tensor representation by decomposing the computation. In this work we use two modalities and this allowed us to apply this technique without requiring the usage of the method proposed in [Liu et al. (2018)].

3 Related literature

This section investigates the approaches taken to solve the document image classification problem. Works prior to [Krizhevsky et al. (2012)] are often based on two approaches - template matching and image descriptors. The sections 3.1 and 3.2 summarize those two approaches while the following sections present newer solutions.

3.1 Template matching and hand-crafted features

Template matching is the process in which a document is being matched to a set of templates through a similarity function. The templates are hand-crafted. An example of this technique can be found in [Sato and Cipolla (1999)].

3.2 Image descriptors

Handcrafted features and templates have been found to be expensive to construct and do not generalize well to new problems. Image descriptors are an approach used to make local decisions at every pixel to decide whether there is an image feature of a given type at that point or not. An example of this can be found in [Bay et al. (2008)].

3.3 Image classification

The majority of recent literature on the topic approaches the problem of image classification through the use of supervised machine learning in order to learn important visual features of the various image categories. After the pioneering work of [Krizhevsky et al. (2012)] convolutional neural networks and transfer learning [Yosinski et al. (2014)] have been applied to solve the document image classification problem - a combination that has shown to generalize to many other computer vision classification problems as shown in [Kornblith et al. (2019)]. The majority of recent literature uses convolutional neural networks for the problem as their performance has been consistently better than that of other models [Harley et al. (2015)]. In [He et al. (2015)] it was shown that features learned by a convolutional neural network outperform handcrafted features given they are trained on sufficient amounts of data. This has also been confirmed for the problem of

document image classification in [Harley et al. (2015)].

3.4 Transfer learning

Transfer learning for document image classification has also drawn criticism in [Tensmeyer and Martinez (2017)] by highlighting the difference in the domains of the images between the ImageNet dataset [Krizhevsky et al. (2012)] and the IIT-CDIP dataset introduced in [Harley et al. (2015)]. Indeed, only the category Web from ImageNet contains images similar to documents [Afzal et al. (2015)]. Nonetheless, transfer learning has been shown to both improve the accuracy of a model and reduce the training time [Tensmeyer and Martinez (2017)].

3.5 Data augmentation

We want to create a classifier which is able to perform well on real-life images. These images may be scanned which may affect the rotation, translation or clarity of the image. We want to explore what possibilities exist to do that in a way that fits our resource constraints. One such technique we experiment with is RandAugment as specified in [Cubuk et al. (2020)] which recently managed to improve the state of art accuracy in combination with the EfficientNet architecture [Tan and Le (2019)] on the ImageNet dataset.

The main goal of data augmentation is to find the optimal policy by which to apply those transformations to images. The policy specifies with what probability and magnitude to apply an operation. In AutoAugment [Cubuk et al. (2019)], a reinforcement learning algorithm is used to find the optimal policy, however that approach proved to be computationally expensive. In RandAugment, there are only 2 hyper parameters that are used:

- N - the number of transformations to apply consecutively
- M - the magnitude of each of the operations

The operations RandAugment uses are displayed below.

1. *identity*
2. *rotate*
3. *posterize*
4. *sharpness*
5. *translate-x*
6. *translate-y*
7. *auto-contrast*
8. *solarize*
9. *contrast*
10. *shear-x*
11. *shear-y*
12. *equalize*
13. *color*
14. *brightness*

Furthermore, [Cubuk et al. (2020)] show that RandAugment performs on-par and sometimes even better than other state of the art data augmentation approaches.

3.6 Text classification

Text classification is the process of classifying a text document by using the textual information in it. In our situation, many of the words in our datasets can be abbreviations or may have little meaning outside the context of the dataset. Similarly to [Audebert et al. (2020)], we use the OCR (optical character recognition) engine Tessaract [Kay (2007)] which may sometimes produce faulty results. Current approaches to text classification aim to solve the problem by learning how to map the words in a text corpus into a high dimensional space where similar words are closer to each other. This process is known as *embedding*. One technique to construct such an embedding for the purpose of this paper is the fastText [Bojanowski et al. (2016)] model used in [Audebert et al. (2020)]. fastText takes a sequence of words as input and maps every word to a set of vectors. Word embeddings generated by fastText are used to create document embeddings. There are several approaches to create document embeddings. The easiest way to do so is to average the word vectors in a document. This approach was first shown in [Kenter et al. (2016b)] and has proven to be successful.

3.7 Multi-modal learning

Previous work shows that use of a multi-modal approaches improves the accuracy of the model when compared to approaches which use a single modality [Audebert et al. (2020), Engin et al. (2019)]. [Engin et al. (2019)] combine the image modality (image of document) and the text modality by extracting the latter through OCR. In structured

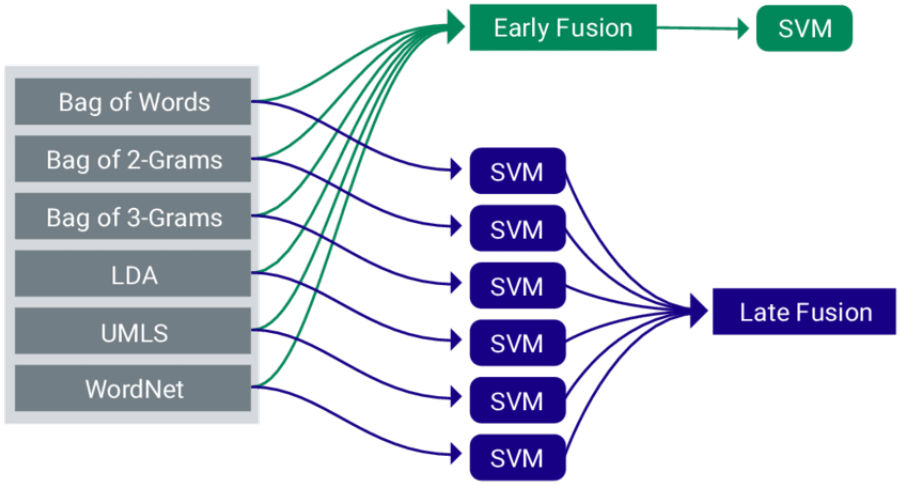


Figure 3.1: Overview of fusion approaches. Source - [Ebersbach et al. (2017)]

documents, text is strongly correlated with the correct category assigned (see table 5.3).

[Audebert et al. (2020)] estimate a 7.6% absolute improvement (9% relative) in accuracy when there's perfect fusion between text and image on the Tobacco-3482 dataset on resolution 384x384.

Multi-modal fusion varies by the use of early, late or hybrid fusion of modalities [Snoek et al. (2005), Baltrusaitis et al. (2017)]. In early fusion features extracted from each modality are concatenated together and a model is trained to make predictions from those features. In late fusion a model is trained on the predictions made by models trained on each modality. Examples of early fusion for document image classification can be found in [Audebert et al. (2020)], while late fusion can be found in [Das et al. (2018)] through the stacked generalization (stacking) technique introduced in [Wolpert (1992)]. An example of a hybrid fusion model is presented in [Vielzeuf et al. (2019)] where the fusion type is being dynamically chosen on a scale between fully early and fully late fusion.

A notable disadvantage of late fusion is that inter-modal interactions are not being included. The final fully connected layer of figure 3.1 learns to guess which model to trust more instead of learning how to interpret the features of the modalities.

An overview of early and late fusion is presented in figure 3.1.

3.8 The effects of scaling

Due to the abundance of documents and their availability in various resolutions, we investigate how we may take an advantage of that amount of data. There are approaches to improve the accuracy of a neural network though the use of architecture search [Zoph and Le (2017), Zoph et al. (2018), Jin et al. (2019)]. In this research, we decided to focus on investigating the compound scaling approach on an architecture discovered through the use of architecture search. Through *compound scaling* [Tan and Le (2019)] a relationship between the width and depth of a neural network and the resolution of images is found and this relationship provides a technique to find the optimal scaling coefficients for network depth, width and image resolution. This relationship is illustrated in equation 3.1. In this equation, ϕ is a user specified coefficient which controls the resources available for model scaling. Authors performed a grid search and proposed the parameters $\alpha = 1.2$, $\beta = 1.1$, $\gamma = 1.15$.

$$\begin{aligned} depth : d &= \alpha^{\phi} \\ width : w &= \beta^{\phi} \\ resolution : r &= \gamma^{\phi} \\ s.t. : \alpha.\beta^2.\gamma^2 &\approx 2 \\ \alpha \geq 1, \beta \geq 1, \gamma &\geq 1 \end{aligned} \tag{3.1}$$

To our knowledge compound scaling has not been applied to the problem of document image classification. The approach proposed in [Xie et al. (2019)] uses, in addition to compound scaling, a combination of supervised and unsupervised learning to further improve the performance of the model. This is a technique called *semi-supervised learning*. While this approach may find its uses for the problem of document image classification, it requires a substantial amount of unlabelled data to lead to significant improvements in the accuracy of the model. [Xie et al. (2019)] use 300M unlabelled photos which leads to 2% improvement in accuracy on the image classification problem on the ImageNet dataset.

For this research we primarily focus on compound scaling on the EfficientNet architecture [Tan and Le (2019)] and investigate its effects on the model performance. EfficientNet is

a convolutional neural network that comes with 8 different versions specified as B0 to B7 with the first version corresponding to a resolution of 224x224. Later versions are scaled according to equation 3.1. A useful comparison between EfficientNet and other state of art architectures is available in figure 3.2.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**77.3%**	**93.5%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.2%**	**94.5%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.3%**	**95.0%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.7%**	**95.6%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**83.0%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.7%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.2%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Figure 3.2: EfficientNet parameters - source [Tan and Le (2019)]

3.9 Attention mechanisms

Attention mechanisms are a technique that attempts to mimic the way humans perceive information by focusing on specific features with weighted importance. For example, when learning to differentiate between the sports ice skating and skiing, a human will learn to focus on the part of the image where the legs of a human are while paying less attention to the other parts of the image. Attention mechanisms have been used for both intra modality (within a single modality) and inter modality (across several modalities) [Guo et al. (2019)]. For single modality, it can be used to select salient features within a modality or balance the contribution of several modalities during multi modal fusion [Guo et al. (2019)]. Examples of intra modality applications can found in classification of ads [Åberg (2018)]. Attention mechanisms have been used in video classification [Long et al. (2018)], image captioning [Xu et al. (2015)], sentiment analysis and image classification

[Bai et al. (2018)], among other practical applications [Guo et al. (2019)].

3.10 Intra-domain learning

Through the use of transfer learning, a technique called intra-domain learning aims to improve the speed of training. This technique uses the neural network weights trained on the whole document (often called holistic approach) and transfers them to smaller models which are then fine-tuned on fixed regions of the document. The main advantage of this approach is that general features such as writing style and font are learned from the whole document while fine-tuning can occur for region-based classifiers. However, there are problems with this approach:

- It assumes that the weights learned on ImageNet transfer well to the problem of document image classification, while [Kornblith et al. (2019)] show that when a task is fine-grained the weights of a network trained on ImageNet do not transfer well.
- The regions of the sub-models are fixed, it is not clear how their number and locations should be chosen.

4 Data

4.1 Overview

In this section we describe and analyze the datasets that will be used in this study.

4.1.1 RVL-CDIP dataset

This dataset has been made freely available online[1]. It consists of 16 document categories and is a subset of the larger IIT-CDIP dataset which is created from the litigation documents against tobacco companies. It consists of 400,000 images split into 16 classes with 25,000 images per class and split into 3 groups - 320,000 training images, 40,000 validation images, and 40,000 test images. This dataset is often used to initialize the weights of a neural network which is then further fine tuned to a particular task.

4.1.2 Tobacco-3482 (T3482) dataset

In this thesis we decided to use the Tobacco-3482 dataset[2], a subset of the RVL-CDIP dataset and commonly used by researchers in the field to benchmark the performance of various approaches. This is a good fit for our purposes because it allows us to experiment efficiently due to the small size of 3482 document images across 10 document types (categories). Examples of Tobacco-3482 are shown in figure 4.1.

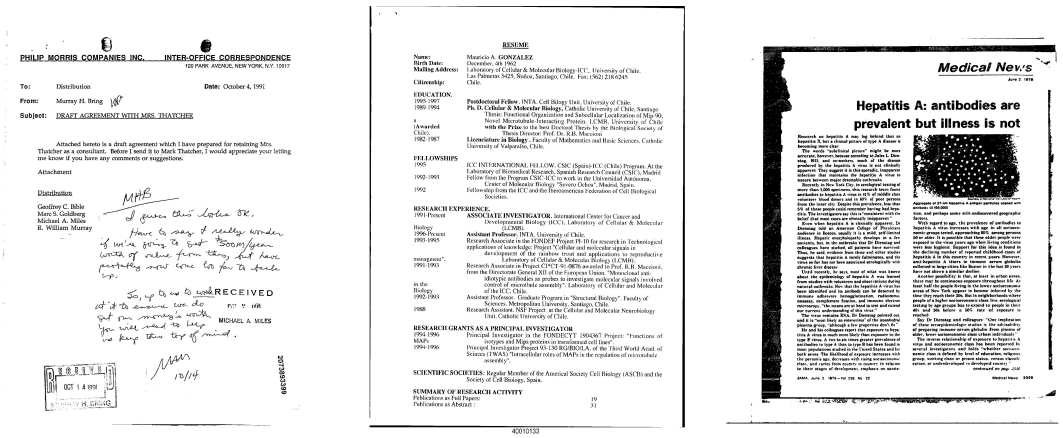


Figure 4.1: A memo, resume and scientific article from the Tobacco-3482 dataset

[1] https://www.cs.cmu.edu/%7Eaharley/rvl-cdip/
[2] https://lampsrv02.umiacs.umd.edu/projdb/project.php?id=72

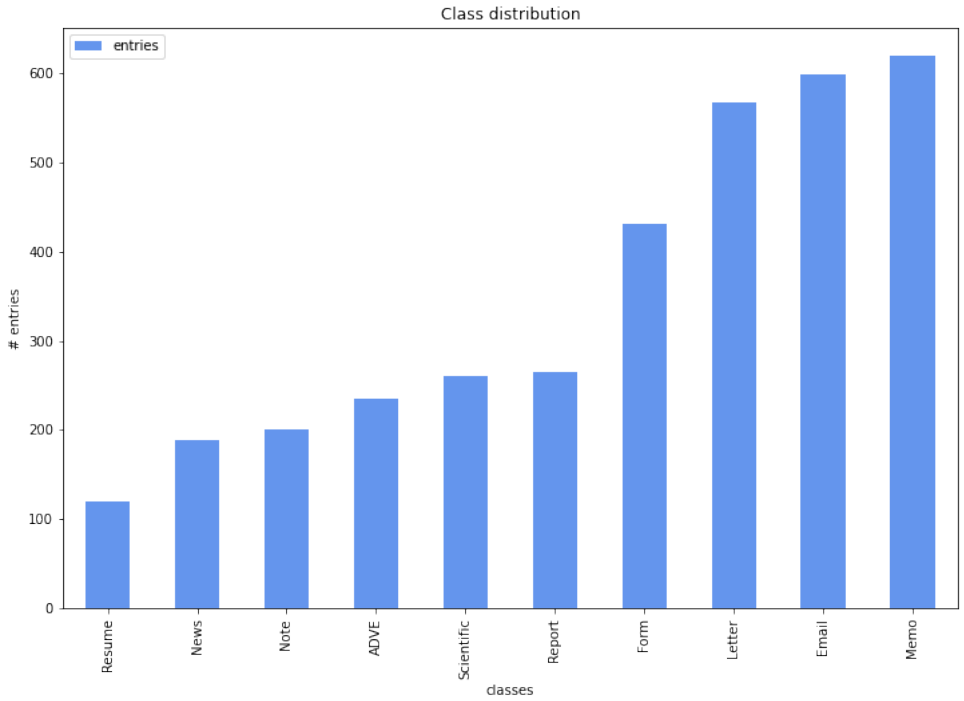


Figure 4.2: Distribution of classes in the Tobacco-3482 dataset

4.1.3 INTERNAL dataset

We have collected a dataset of 15,258 *.pdf* document images - *INTERNAL* across 66 categories. We use this dataset to observe how well our best performing model works on real life data. This dataset contains proprietary images, therefore we report only the results of our analysis on it. Despite not publishing the dataset, we offer guidelines on how to construct a training pipeline for a real world scenario where the data may require pre-processing. The images in INTERNAL contain more geometric figures than the other datasets. An example of 2 types of the documents present in *INTERNAL* can be seen in figure 4.3. The distribution of the images per class is shown in figure 4.4.

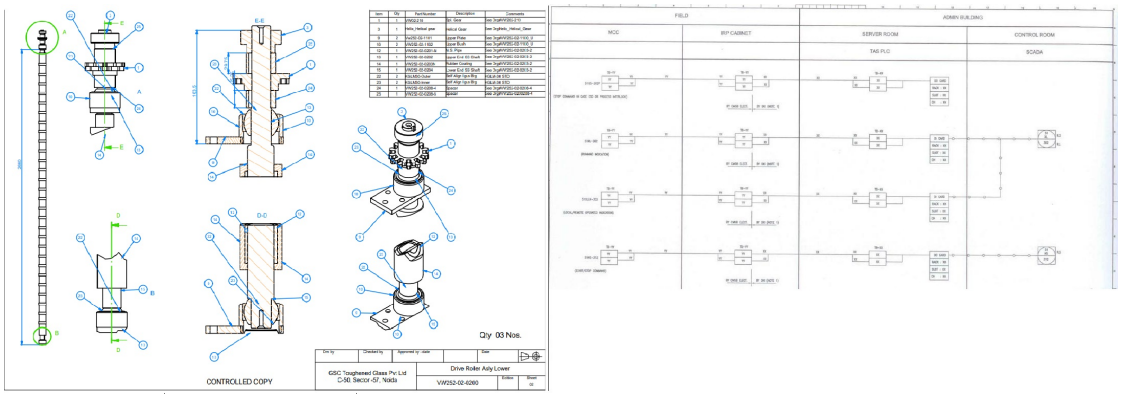


Figure 4.3: A .dwg diagram and a loop diagram, 2 of the classes of the INTERNAL dataset. Source -

The dataset was prepared by converting all *.pdf* pages into images by using a custom made tool[3]. The document images, among relevant images, also contain pages containing content table pages, introduction pages or other information. This means that some of the documents in the dataset will be irrelevant but nonetheless labeled as belonging to that category - for example, a content page can never be viewed as a diagram. This flaw can be addressed in future research by for example, creating a small dataset of images which surely belong to a class, and then training a classifier on them. The classifier can be used to make a binary prediction whether a given image should be included in the final dataset.

[3]https://github.com/ch-hristov/pdf2imgs

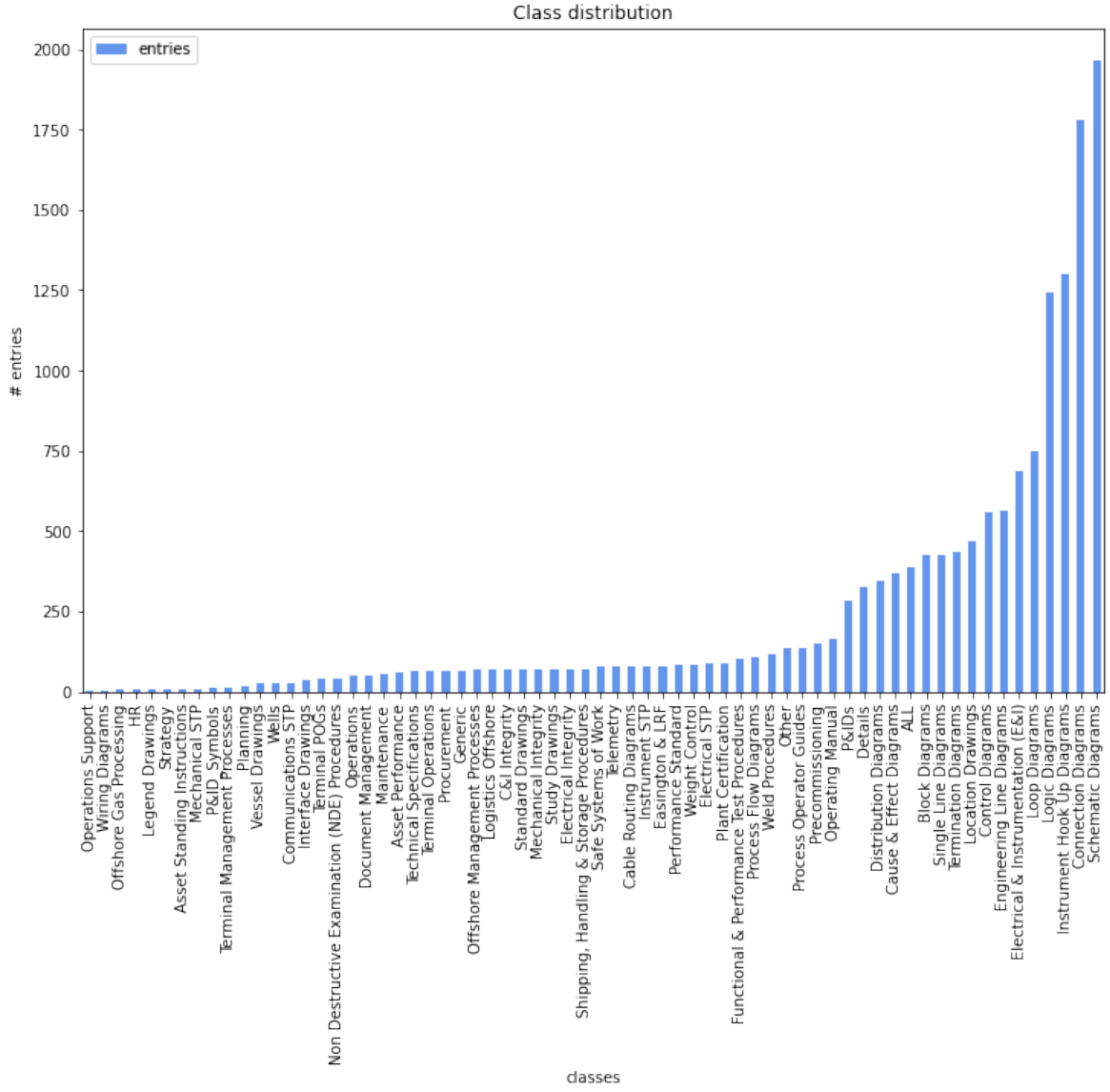
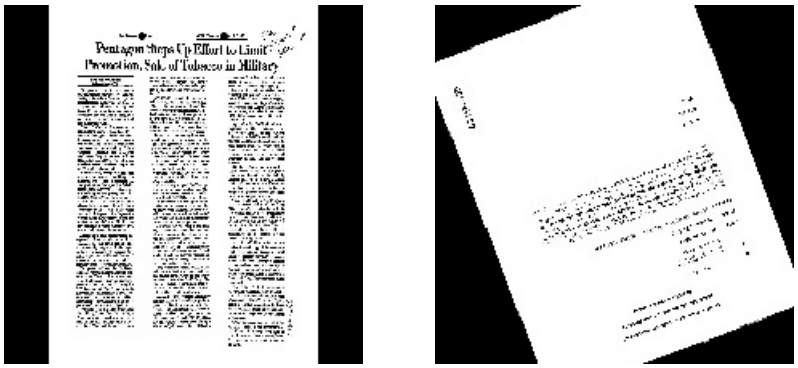


Figure 4.4: Distribution of number of items per class in the INTERNAL dataset



(a) Padded image example

(b) Padded and augmented image example, only used in section 6.3

Figure 4.5: Preprocessed images from the T3482 used in this work

4.2 Data preprocessing

4.2.1 Images

All images used for training in this thesis are preprocessed in the following way:

1. Resize and pad - into a given resolution and padded with black color if necessary. This is necessary because some of the original images have different resolutions, therefore to keep the aspect ratio of the image, some of the images are padded with black space as shown in figure 4.5. This has been shown to not affect the accuracy of models as they simply learn that the black space does not offer any value to a learner. An example of such an image which has been resized, padded and normalized and augmented from the T3482 dataset is available in figure 4.5.

2. Center - After resizing and padding the image we subtract the mean pixel values of the training images from all images to center the data. This preprocessing step is the same as the one taken in [Afzal et al. (2017)].

4.2.2 Text

All text extracted from the images, after extracting it through Tessaract OCR, is processed in the following way (see Appendix for detailed code snippet):

1. Non-letters are removed

2. Text is lemmatized and tokenized
3. Non meaningful words are removed - stop words

In listing 1 we show example text extracted through OCR.

```
0    citi week septemb guid urban nightlif iaadaypl...
1    arthur j steven dp la cewsn date c oo eeeeee l...
2    de de de transmiss report x e jul id start tim...
3    hoel john hoel john sent wednesday decemb pm m...
4    februari mr w r lybrook refer attach letter mr...
```

Listing 1: Example OCR extracted text from the T3482 dataset

4.3 Data augmentation

It has been shown that augmenting the input data can create a significant improvement in the accuracy of the model, with the *shear* operation being the most useful augmentation on the RVL-CDIP dataset [Tensmeyer and Martinez (2017)]. [Tensmeyer and Martinez (2017)] use a CNN for the problem of image classification on ImageNet and presents results on the best way to represent the data. Research suggests that combination of grayscale (G) and dense-SURF (S) is the best way to represent an image [Bay et al. (2008)]. For this research we experimented with RandAugment.

4.4 Class imbalance

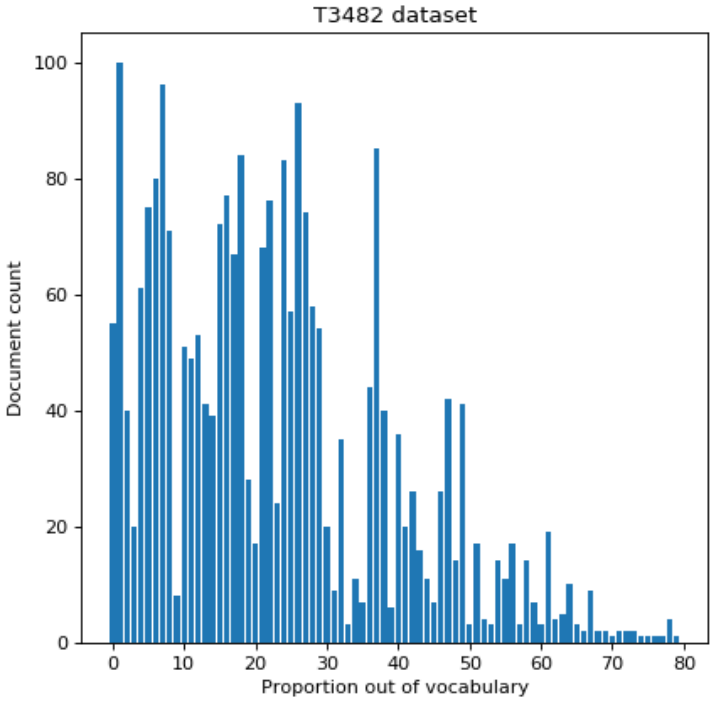
While the RVL-CDIP dataset is balanced, the Tobacco-3482 and the INTERNAL datasets are unbalanced. The distribution of classes of the Tobacco-3482 dataset can be seen in figure 4.2. Machine learning classifiers may develop a bias towards classes in the training set which are more common and that can cause a classifier to overfit. Additionally, it can be more difficult to compare new approaches to multi modal fusion when the distribution is not balanced. In reality, it would be difficult to know the real distribution of the document image categories for most document image classification tasks. In order to address the imbalance, we use two methods - for the Tobacco-3482 dataset we use a

balanced version of the data set. This balancing procedure is reviewed in section 5.2.2. For the INTERNAL dataset we report measurements which we believe are representing the performance of the model as well as possible.

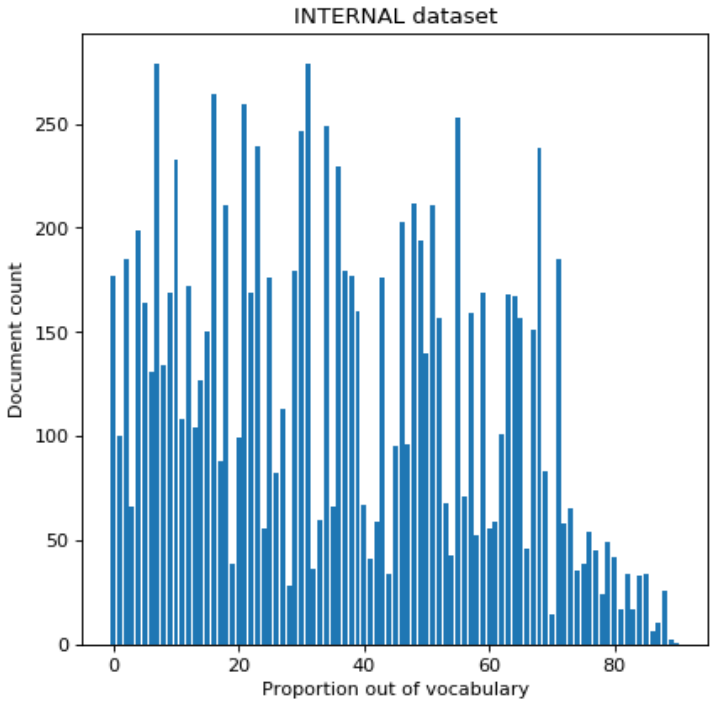
4.5 Out of vocabulary words

A good way to predict how difficult the INTERNAL dataset would be for a text classifier is to compare the OOV (out of vocabulary) count distribution between T3482 and INTERNAL. We do not perform this analysis for RVL-CDIP as the T3482 dataset comes from a common superset as RVL-CDIP and we assume T3482 should provide a good understanding of the distribution for RVL-CDIP as well. Similarly to [Engin et al. (2019)] and [Audebert et al. (2020)], we are concerned about low performance of the text model due to faulty results by the OCR. For this task, we first extract the text for INTERNAL and T3482 by using Tessaract OCR [Kay (2007)]. We then use a pretrained set of word vectors[4] over which we check each document word for each document. The results of this analysis are shown in figure 4.6 for the T3482 and INTERNAL datasets respectively. What is immediately visible is that the INTERNAL dataset seems to have a much higher proportion of values which are OOV. This could be due to the fact that some of the documents contain Dutch words or due to inability of the OCR engine to detect the words correctly.

[4]https://fasttext.cc/docs/en/english-vectors.html



(a) OOV words T3482



(b) OOV words INTERNAL

Figure 4.6: Comparison OOV distribution between T3482 and INTERNAL

5 Methodology

5.1 Research questions

We consider the document image classification problem a subset of the image classification problem. The error rate of humans on the ImageNet dataset has been reported to be 5.1% in [Russakovsky et al. (2015)]. The state of the art accuracy to this date on the ImageNet dataset is 87.4% by the Noisy student model by [Xie et al. (2019)]. The state of art accuracy for the document image classification problem is 92.21% on the RVL-CDIP data set in [Das et al. (2018)] by using stacked generalization. This result, however, is not applicable in this research as it uses a network pretrained on RVL-CDIP that is later fine tuned to the T3482 dataset. In reality, a larger labelled dataset often does not exist from which to perform such an initialization step and there is no guarantee that the weights from RVL-CDIP would transfer well on our own data set. Therefore we fall back to the results which are obtained through an ImageNet pretrained weight initialization.

The research questions for this thesis are:

1. How does EfficientNet compare to state of art architectures? Is compound scaling applicable to the problem of document image classification?

2. Should data augmentation be applied to document image classification? Does the RandAugment [Cubuk et al. (2020)] data augmentation technique improve the robustness of a classifier?

3. How do tensor fusion methods [Zadeh et al. (2017)] compare to the more well known methods of early and late fusion on the balanced data split for T3482? Does the model configuration found do well on INTERNAL? Does the best method we find on the balanced data set split perform the best on the all data split?

5.2 Research methodology

All experiments performed in this work are used on two dataset splits, we refer to them as *all data* and *balanced*. The *all data* split is used because we're interested in obtaining the best possible classifier which can be used in real-life scenarios, therefore we want to

use as much data as possible for learning. The second data set split is used to compare our approaches to previous work and this approach balances the training and validation set so that there are an equal number of images for every class.

5.2.1 Data set split - all data

For this approach we select 70% of the data as the training set, 10% as the validation set and 20% as the test set. This is shown in figure 5.1. The data set is shuffled before any selection is done. In some cases only 10% (shown in red) of the data is used for test set. In this work we decided to use 20% as the size of the INTERNAL dataset where this split is used isn't large enough to use only 10% of the total dataset.

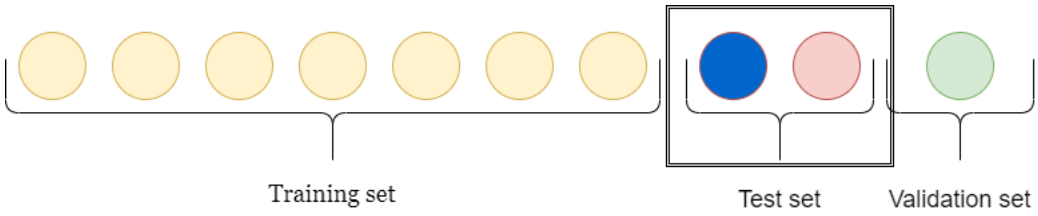


Figure 5.1: Dataset split - each circle represents 10% of the total dataset

5.2.2 Data set split - balanced

In order to get measurements which are comparable to other approaches we use the method shown in [Afzal et al. (2017)] which is comparable to the methods used in previous research [Kang et al. (2014)] to balance the dataset by randomly taking N = 100 samples from each class. 80% of those N items are used as the training set while the remaining 20% are used for validation set. The remaining items which are not part of those N samples per class are used as the test set. The baseline for T3482 defined in [Afzal et al. (2017)] is used and is available in table 5.1. We report the results for a split size of 100 and we average the results over 10 different splits similar to [Afzal et al. (2017)]. This approach is not directly comparable to a model which was initialized with weights from a model trained on a larger data set such as the RVL-CDIP. The results from such an initialization are illustrated in table 5.2.

Architecture	Top 1 Accuracy
VGG16 [Afzal et al. (2017)]	77.52%
AlexNet [Afzal et al. (2017)]	75.73%
GoogLeNet [Afzal et al. (2017)]	72.98%
Resnet-50 [Afzal et al. (2017)]	67.95%

Table 5.1: Dataset: T3482, **Image** model accuracy, resolution (224x224), ImageNet weight initialization

Architecture	Top 1 Accuracy
VGG16 [Afzal et al. (2017)]	91.01%
AlexNet [Afzal et al. (2017)]	90.04%
MobileNetV2 [Audebert et al. (2020)]	84.5%

Table 5.2: Dataset: T3482, **Image** model accuracy (224x224), RVL-CDIP (excl. T3482) weight initialization

5.2.3 Text model

To train the text classifier, we use an approach taken in [Audebert et al. (2020)]. To create word embeddings we use the fastText model (for exact software tool version see the Appendix). As input to the **Text** model we use the word embeddings, averaged for each document. The exact architecture of this approach is shown in figure 5.2. As input we use word embeddings of size 300. This network was initialized using He's initialization [He et al. (2015)]. The difference between [Audebert et al. (2020)] and our approach is that the last dense layer is of size 100 rather than 128. This is because the **Image** model performed slightly better during our hyper parameter tuning process with a final layer of size 100 for the INTERNAL dataset. We use that size for the **Text** model rather than 128 in order to avoid using two different sizes when training the multi modal fusion networks presented in the experiments chapter.

Author	Embedding	Architecture	Accuracy
[Audebert et al. (2020)]	fastText	MLP	70.08%
[Audebert et al. (2020)]	fastText	CNN 1D	73.9%

Table 5.3: Text model results on the T3482 dataset. Note: [Audebert et al. (2020)] uses cross validation with the same train/test/validation set sizes

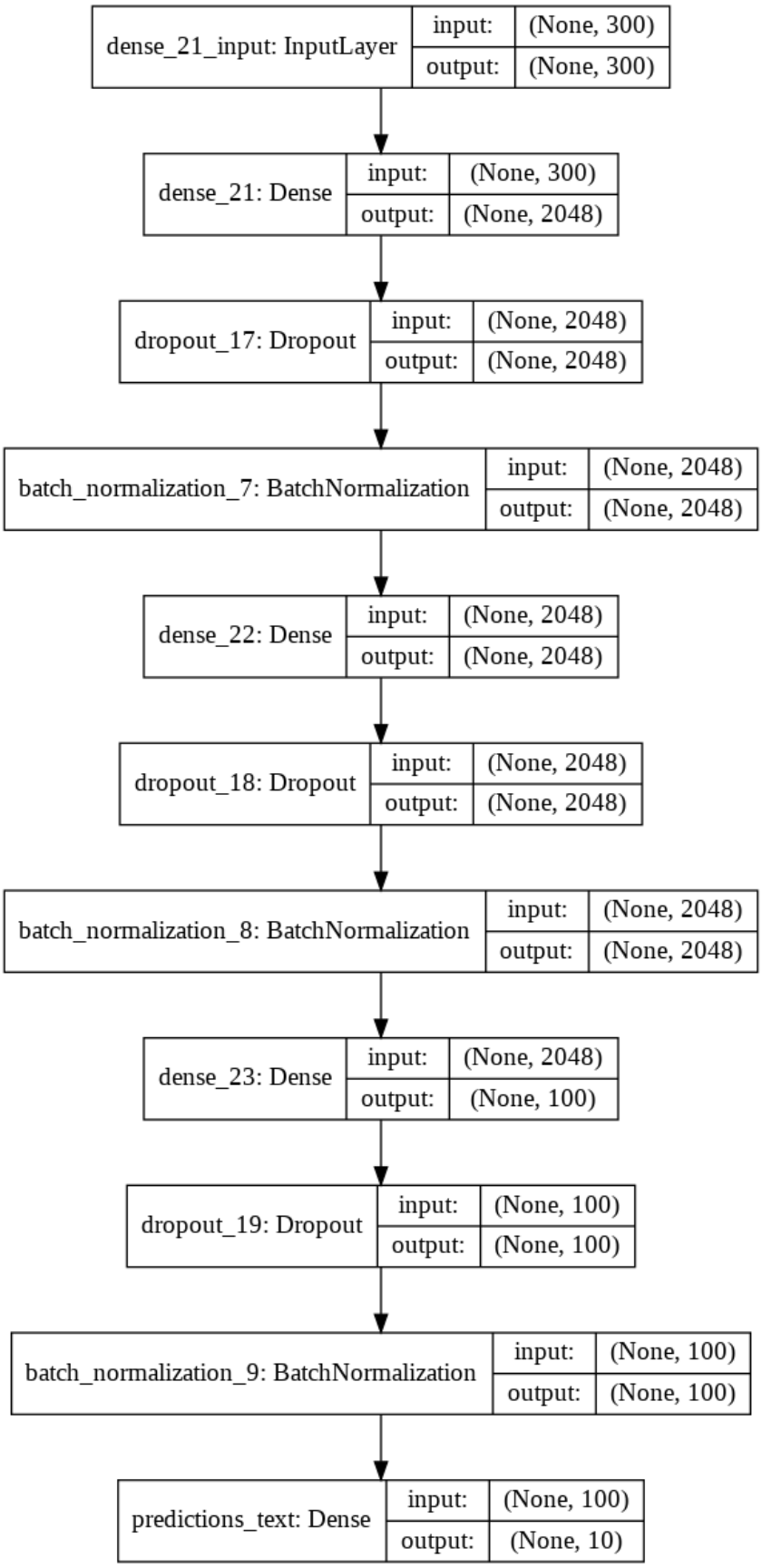


Figure 5.2: Text model used in this work. Initially proposed in [Audebert et al. (2020)]

5.2.4 Data set split for multimodal fusion models

It is not clear what is the best way to split the data in order to train a multimodal fusion network. In [Åberg (2018)] half of the validation set is used to train a network that learns from separately trained **Image** and **Text** models. This split is compared with training the fusion network on the same data as both of the models were trained on. In [Åberg (2018)] training on the same data as the two models were trained on shows better performance. We decided to use this approach as our experiments showed that it had better performance than training on half of the validation data or by using an additional 10% of the original set. This is also partially supported by the results found in [Åberg (2018)] on the same matter where a comparison is made between training on original training set and training on half of the validation set.

5.2.5 Early stopping

For T3482 we use early stopping to choose the model which minimizes the validation loss. For the INTERNAL data set we maximize the validation set F1 score.

5.2.6 Baseline analysis

We experiment to see what are the potential improvements from applying multi modal fusion for the INTERNAL dataset. Such an analysis was already done in [Audebert et al. (2020)] for the T3482 dataset. A way to quantify that, is to see how to compare the predictions of the supervised model provided in the fastText package - [Bojanowski et al. (2016)] and the EfficientNet B0 vision model and see in how many cases either one of them, both of them or neither of them are correct on the test set. Note that this measure depends on the models. Better text or vision models may reduce the possible improvement from multi modal fusion. This does not take into account improvements that may occur from finding patterns of bimodal features but it should be enough to tell us whether it is worth investigating the multi modal fusion approaches for our dataset. In table 5.4 we measure those values for the INTERNAL dataset.

The results show a possible improvement of 9.26% through perfect fusion of text and image. Note that these results are obtained with the B0 classifier on resolution 224x224,

INTERNAL	EfficientNet B0	FastText supervised	Either correct	Both correct
Correct	2478	2368	2761	2085
Incorrect	579	689	296	972
Total %	81,06%	77,46%	90,32%	68,20%

Table 5.4: Potential improvement, INTERNAL dataset, resolution 224x224, all data data split, test set samples $N = 3057$

therefore the vision model will perform better on higher resolution images. This example should convince the reader for the case of multi modal fusion for this dataset.

6 Experiments

6.1 Environment

6.1.1 Hardware

We perform experiments using an Azure GPU virtual machine of the type NC6_Promo (6 vcpus, 56 GiB memory, Windows 10 Pro). This environment is equipped with the Tesla K80 GPU on which all training was done.

6.1.2 Software

There are two important dependencies for this thesis:

1. **Image** model - The neural networks in this thesis have all been constructed using Tensorflow 1.14 and Keras 2.3.1.
2. **Text** model - We use the fastText model to create word embeddings. We then use the architecture specified in figure 5.2.

The full list of packages is available in the Appendix.

6.2 How does EfficientNet compare to VGG-16?

In this section we perform an experiment to compare EfficientNet to the state of art architecture VGG-16. We chose to compare it to VGG-16 because VGG-16 has consistently shown that it performs well on the problem of document image classification - [Afzal et al. (2017), Das et al. (2018), Audebert et al. (2020)]. Additionally, this architecture has publicly available weight configuration trained on the RVL-CDIP dataset which decreases the amount of training we have to do. This experiment uses the compound scaling technique as explained in section 3.8. We train EfficientNet B3 and compare it to the VGG-16 architecture. There are two reasons why we choose the B3 architecture instead of using a larger model.

1. The VGG-16 model using this configuration has trainable parameters equal to 18,883,198 while the EfficientNet B3 model has 10,696,232 trainable parameters. This makes the two architectures arguably comparable in the number of parameters.

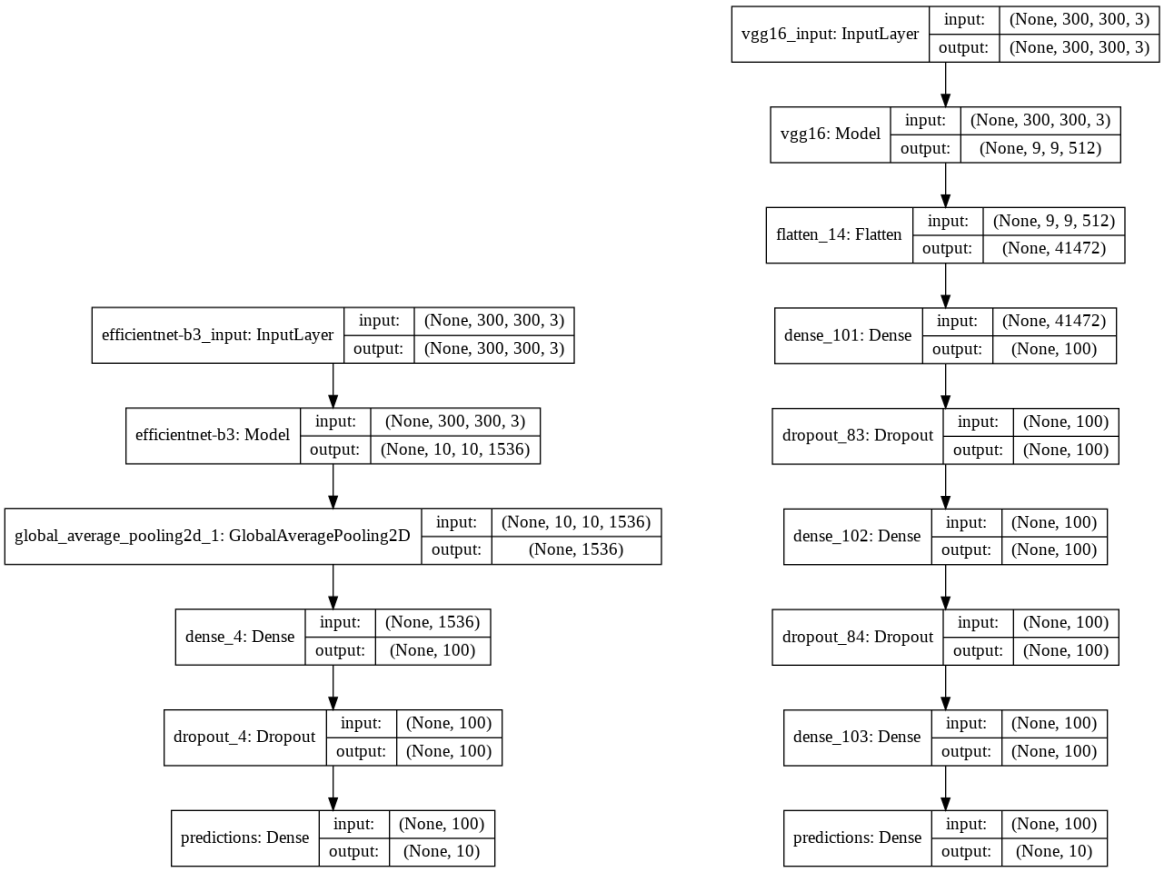
2. The fact that the Tesla K80 video card runs out of memory when we scale up above EfficientNet B3 and we would like to keep the experiments accessible on hardware that is easily available.

We use the architectures shown in figure 6.1. We also ran the experiments by replacing the final Global Average Pooling 2D by a flattening layer for the EfficientNet architecture but that resulted in lower accuracy. This is perhaps because Global Average Pooling 2D is a dimensionality reduction technique and we believe that adding this pooling layer results in less over fitting. It was employed in the same manner in [Audebert et al. (2020)]. The implementation of EfficientNet we use is open source[5].

6.2.1 Hyper parameter optimization

We run a grid search on suitable values for the dense layer sizes for both the VGG-16 and EfficientNet B3 architectures. This helps us to find good hyper parameter values, a suitable drop out rate and to investigate what the effect is of adding a dense layer that encodes a deep representation of the image (embedding).

[5]https://github.com/qubvel/efficientnet



(a) The model used in this work - EfficientNet B3 as base model, inspired by Audebert et al. (2020)

(b) VGG-16 as base model - Afzal et al. (2017)

Figure 6.1: Image model architectures compared in this chapter

6.2.2 Optimizer

As figure 6.2 shows, a reasonable conclusion would be to use the *Adam* optimizer. The encoder layer size can be changed but it conveniently matches the default size of each word vector in FastText.

6.2.3 The effects of compound scaling

We experiment with the non-scaled version of the EfficientNet architecture to see how it will perform in a situation where resources for scaling of the architecture are not available. That means that the default EfficientNet architecture is not scaled. We compare that to the VGG-16 architecture.

The value of $\gamma = 1.15$ is suggested in the original paper for EfficientNet [Tan and Le

optimizer	learn rate	dropout rate	encoder layer	avg loss	avg acc	avg top-3	avg recall	avg precision
Adam	1.00E-05	0.5	0	0.7914	0.7814	0.9316	0.7595	**0.8184**
Adam	0.0001	0.5	0	0.8647	0.7944	**0.9531**	0.7845	0.8126
Adam	1.00E-05	0	100	0.8924	0.7738	0.9378	0.7538	0.7967
Adam	0.0001	0	100	1.0567	0.7814	0.9326	0.7706	0.7941
Adam	1.00E-05	0.5	100	0.8095	0.7853	0.9335	0.7615	0.8161
Adam	0.0001	0.5	100	0.9493	**0.8049**	0.9503	**0.7968**	0.817
sgd	1.00E-05	0.5	0	0.8488	0.7293	0.9307	0.6793	0.7968
sgd	0.0001	0.5	0	0.846	0.7556	0.9311	0.7281	0.7946
sgd	1.00E-05	0	100	0.8968	0.7111	0.9134	0.6436	0.7824
sgd	0.0001	0	100	0.9122	0.7279	0.9177	0.6963	0.7773
sgd	1.00E-05	0.5	100	0.8672	0.7279	0.9149	0.6228	0.8224
sgd	0.0001	0.5	100	**0.7479**	0.7805	0.9326	0.7365	0.8385
sgd	0.0001	0	0	0.799	0.7642	0.934	0.7329	0.7972

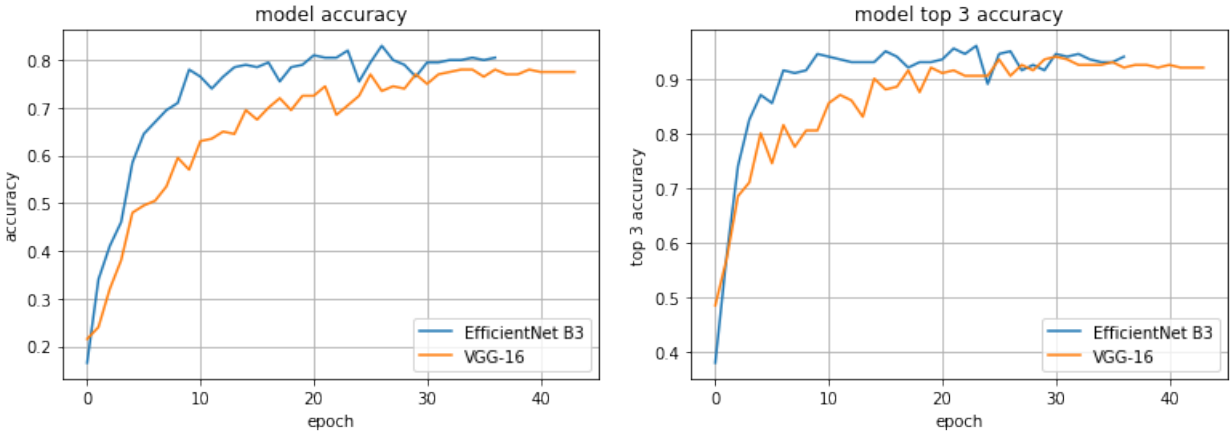
Table 6.1: Grid search results (on the validation set) for T3482 (trained on 70% for the training set) on resolution 224x224 for EfficientNet B0, ImageNet initialization

Figure 6.2: Optimizers tested for this work

(2019)]. We use the value specified in the official Tensorflow EfficientNet implementation[6] for the B3 version. The resolution used for B3 is 300x300. Training with compound scaling is expensive in terms of resources. We managed to train networks with compound scaling up to EfficientNet B3 on a single Tesla K80 video card by decreasing the batch size to as low as 16. This made training slower compared to the B0 version where a larger batch size can be used. The results are presented in table 6.2.

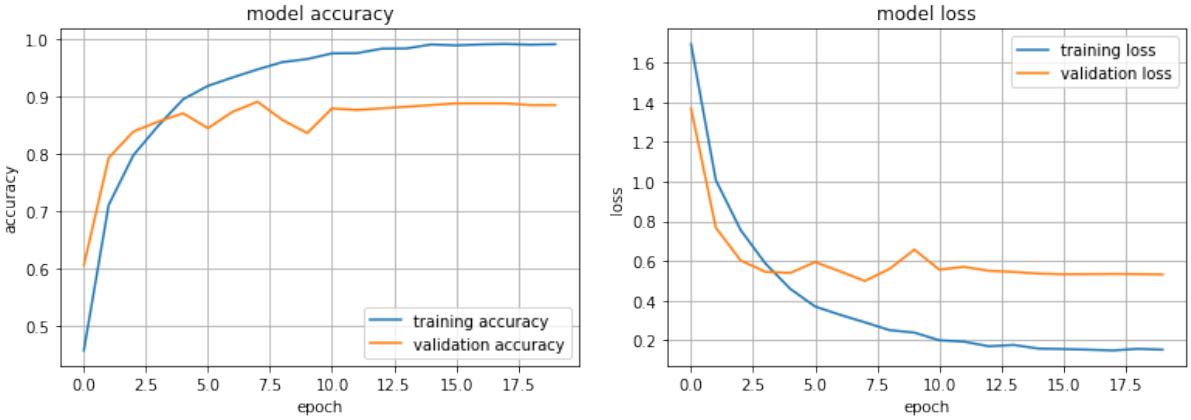
We refer to the vision model as **Image**.

[6]https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/efficientnet_builder.pyL42



(a) Top 1 accuracy **(b)** Top 3 accuracy

Figure 6.3: Validation set, balanced data set split single run, Top 1 accuracy and Top 3 accuracy, Dataset: T3482, Resolution 300x300, EfficientNet vs VGG16



(a) Training and validation top 1 accuracy **(b)** Training and validation loss

Figure 6.4: Balanced baseline, single run validation set measurements, EfficientNet B3 training on T3482

6.2.4 Results

Base model	Top 1 mean acc.	Top 1 acc. std	Top 3 mean acc.	Top 3 acc. std
Text model	0.681708	**0.003155**	0.905261	**0.005214**
Text model [Audebert et al. (2020)]	0.7008	-	-	-
VGG-16	0.733481	0.016209	0.898791	0.014513
VGG-16 [Afzal et al. (2017) 224x224]	0.7752	-	-	-
EfficientNet B3	**0.790894**	0.009495	**0.940290**	0.005630

Table 6.2: Averaged results for T3482 test set (N = 2482) T3482, 10 runs, balanced split

We trained the B3 classifier on INTERNAL and obtained the following results:

Base model	Top 1 mean acc.	Top 3 mean acc.	Training time
Text model (fig. 5.2)	0.7765	0.8861	**<1 min**
VGG-16 (fig. 6.1 b)	0.760341	0.84232	14.001 hrs
EfficientNet B3 (fig. 6.1 a)	**0.8567**	**0.9368**	9.53 hrs

Table 6.3: Results for test set (N = 3057) INTERNAL, 1 run, balanced split

6.3 RandAugment - Efficient data augmentation for real life scenarios

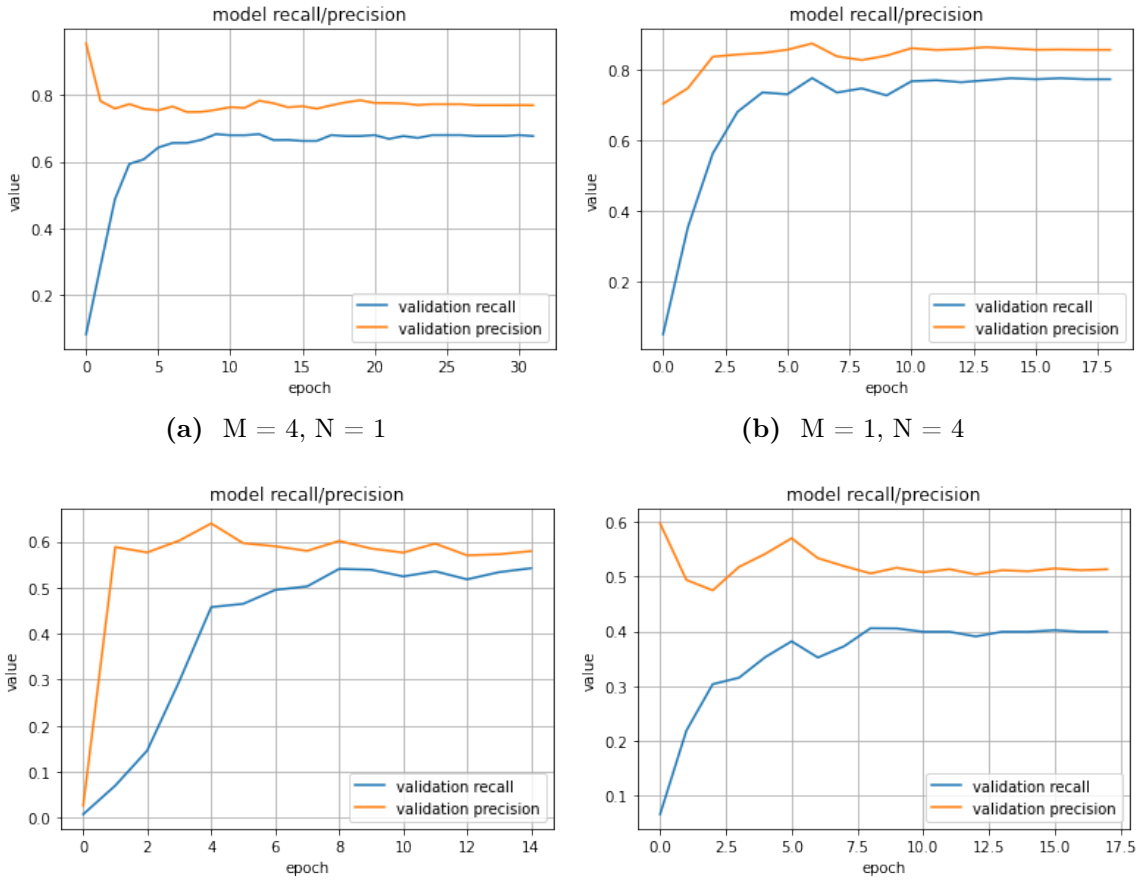
For this experiment, we are answering the question of whether RandAugment can benefit our classifiers.

6.3.1 Setup

For this experiment we modify the input data by adding random transformations in order to make the data more complex for a predictor not trained on noisy data. We use the T3482 dataset with the *all data* split. We use the resolution 224x224 and use EfficientNet B0. The results when the training dataset isn't augmented are shown in figure 6.5. We then perform the experiment with the same validation set but we add the augmented images of the training set to the training set. This doubles the training set size. The results are shown in figure 6.6. These figures show how changing the parameters N and M affects the performance of a classifier which is trained with augmented images and one which is trained without any augmented images.

- N - the number of transformations to apply consecutively
- M - the magnitude of each of the operations

6.3.2 Results



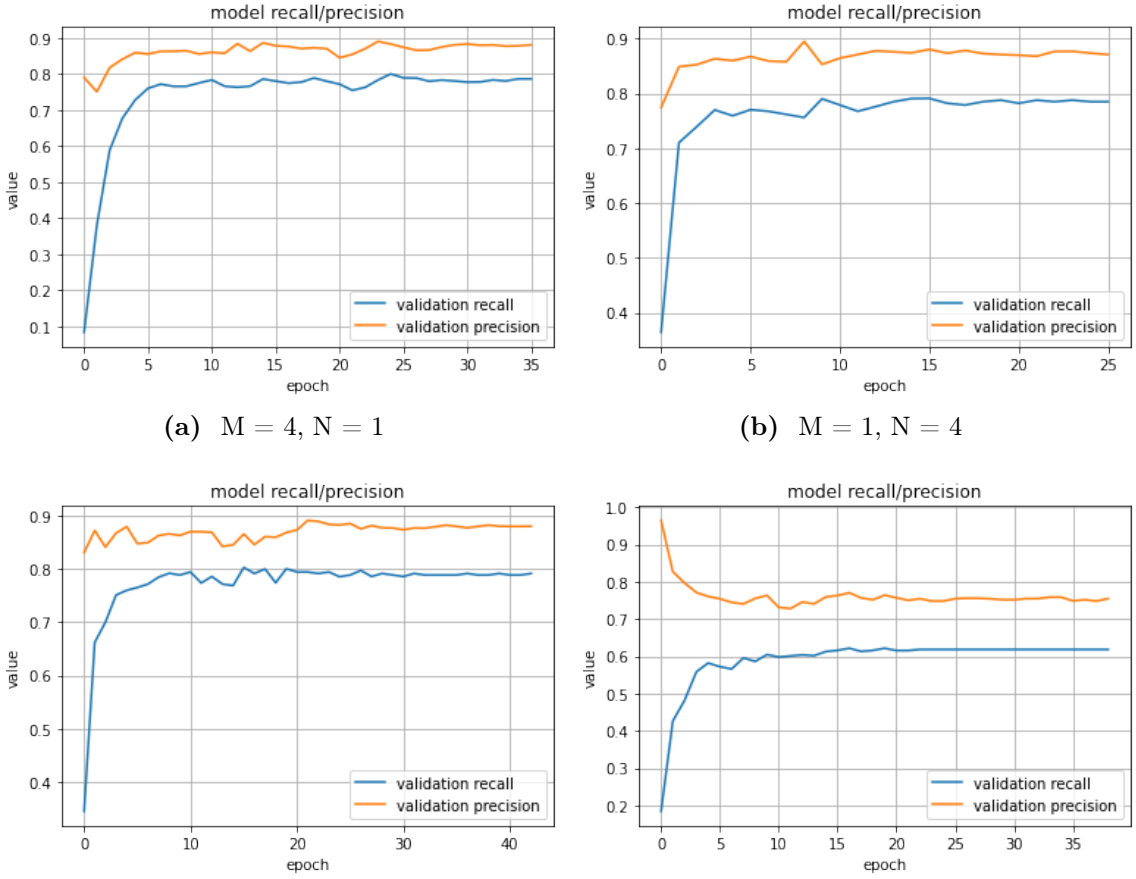
(a) M = 4, N = 1

(b) M = 1, N = 4

(c) M = 4, N = 3

(d) M = 5, N = 5

Figure 6.5: T3482 dataset, all data split, EfficientNet B0, no augmentation on the training set, validation set is replaced with transformed images. The graphics shown are the values on the validation set.



(a) M = 4, N = 1

(b) M = 1, N = 4

(c) M = 4, N = 3

(d) M = 5, N = 5

Figure 6.6: T3482 dataset, all data split, EfficientNet B0, both the validation and training set are augmented. The graphics shown are the values on the validation set. Similar results were observed for the test set as well.

6.4 Multi modal fusion

This section compares three approaches to multi modal fusion. For this experiment, we fix the **Image** and **Text** models and collect measurements for the multi modal fusion approaches described. A short section about the implementation of each of the multi modal fusion methods is included. Afterwards the results of the experiments are presented. All our experiments follow from the idea presented in the Python-style pseudo code below. The exact architectures for each fusion approach can be found in Appendix figures A0.1, A0.2 and A0.3. The code is run for every data set separately.

```
# Choose how to split the data set
#train_xy, test_xy, val_xy = separate_set_into_train( paths_set_original )
train_xy, test_xy, val_xy = separate_afzal( paths_set_original )
# Train text model
model_text = train_text_model(train_xy, test_xy, val_xy)
# Train vision models
vgg_model = train_vision_model('vgg', train_xy, test_xy, val_xy)
effnet_b3_model = train_vision_model('enet', train_xy, test_xy, val_xy)
compare_performance( [ get_predictions(vgg_model),
    get_predictions(effnet_b3_model) ] )
outputs = []
# For INTERNAL - k = 1, for T3482 - k = 10.
for i in range(k):
    for fusion_approach in ['early', 'late', 'tensor']:
        # apply modal fusion
        output = fuse(train_xy, test_xy, val_xy, fusion_approach, vgg_model,
            effnet_b3_model, model_text)

        predictions = get_predictions(output)
        outputs.append(predictions)

    compare_performance( outputs )
```

6.4.1 Early fusion

In this approach we strip the last layer which generates predictions (the softmax layer) of the **Image** and **Text** models and we use the outputs of the final dense layer. This dense layer outputs a compact representation of features for the input. We then concatenate the outputs from the two final dense layers of the **Image** and **Text** models and then add a dense layer to create a representation that combines the two sub spaces, in which the individual representations reside, into a common space.

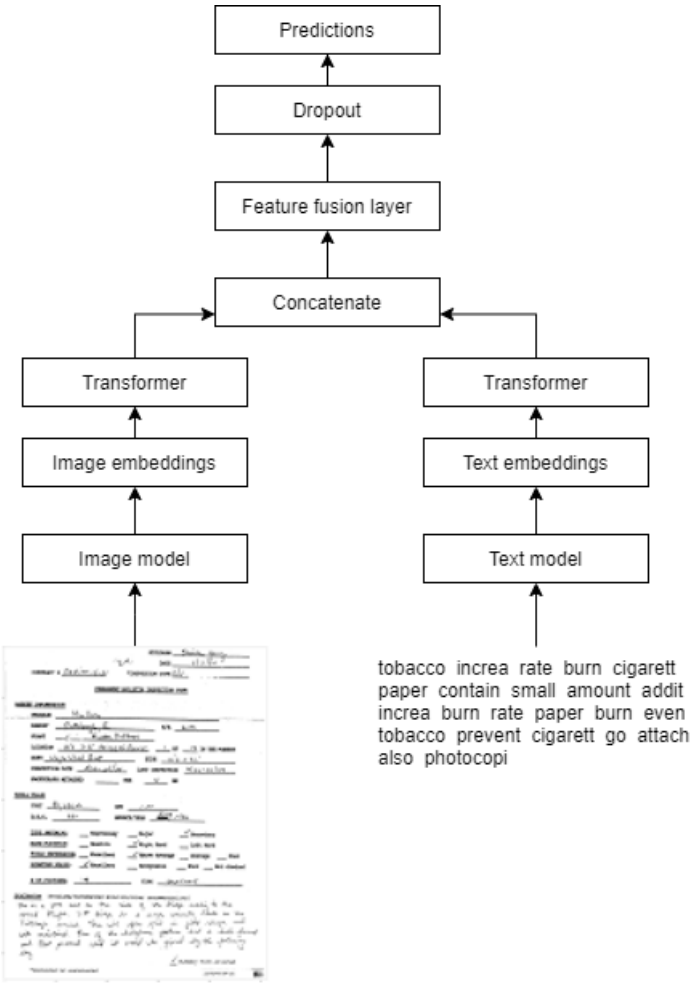


Figure 6.7: Early fusion

Approach

This approach uses a combination of high level features to make predictions. These high level features are gathered from both the text and image learners and are used by a model to make predictions. For the **Text** model, we average the word embeddings generated for each document and use that vector as a document embedding. For the **Image** model,

similar to [Audebert et al. (2020)], we add a dense layer after the last pooling layer of the network in order to create a vector representation for the image. We then apply a learner on the concatenated version of those vectors to produce predictions.

This approach is shown in figure 6.7.

6.4.2 Late fusion

We reviewed several techniques to make predictions by creating a model which is learning from the output of the probabilities of each modality. Note that both of the **Image** and **Text** models are trained with a final layer which uses the softmax activation function. This means that the last layer contains the output probabilities for each class. In this section we create a model which learns from those output probabilities to predict the final output. To do this, we freeze the weights of the trained models, then add a layer that concatenates the output probabilities into a single tensor. This final tensor is passed to either one or two dense layers.

Approach

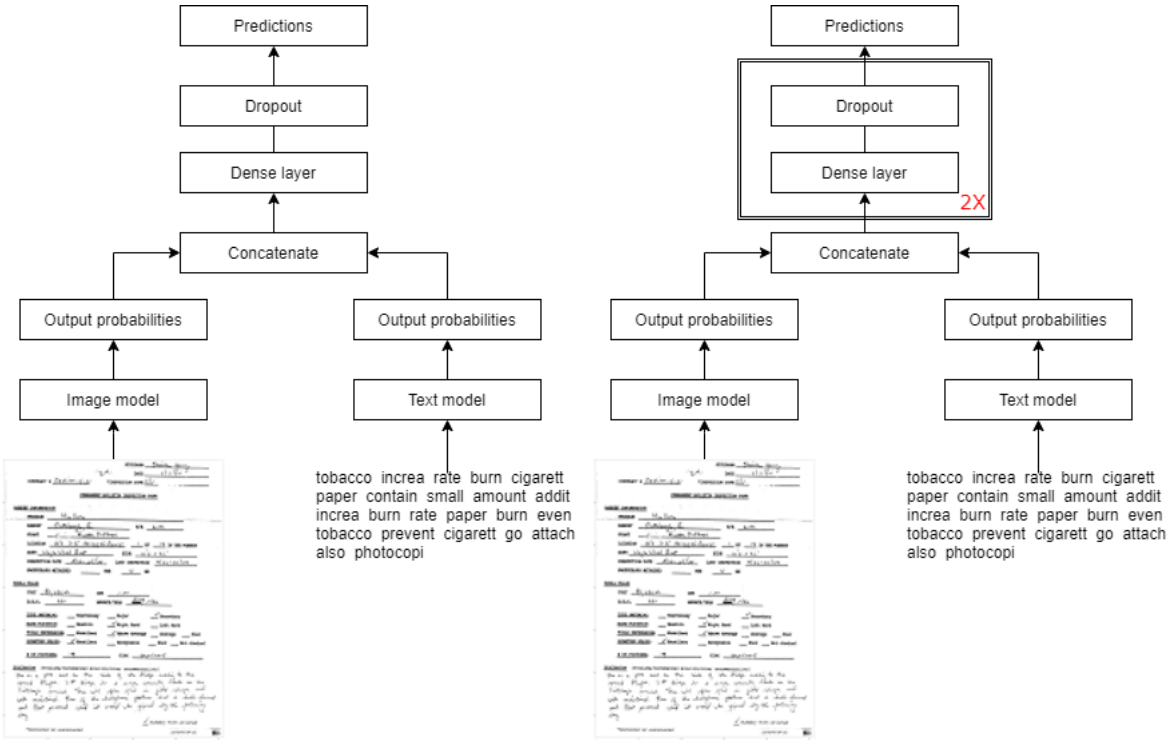
The approaches we test are:

1. Late fusion - a single dense layer learns from the concatenated list of output probabilities for each modality.
2. Late fusion (D) - a network with two dense layers may be able to capture the patterns of the data better. Deeper networks can estimate more complex functions.

An illustration of the two approaches can be seen in figure 6.8.

Hyper parameters

We use some of the parameters provided in [Åberg (2018)] as a base for the grid search in this work. The full list of parameters that are searched and the parameters which perform best for each data set are available in the Appendix.



(a) Single layer late fusion **(b)** Two layer late fusion

Figure 6.8: Late fusion approaches

Selection policies

We considered 2 different policies - *model* based and *heuristic* based. The model based approach consists of creating a model which learns from the output probabilities of each modality. The heuristic approach is a way to predict the final outcome based on a fixed rule. We decided against using rule based policies and decided to focus on model based ones.

6.4.3 Tensor fusion

We adapt an approach which was used for trimodal fusion used in [Zadeh et al. (2017)]. In our work we are working with two modalities. This approach involves computing the outer product of the extracted features. We illustrate our approach in figure 6.9.

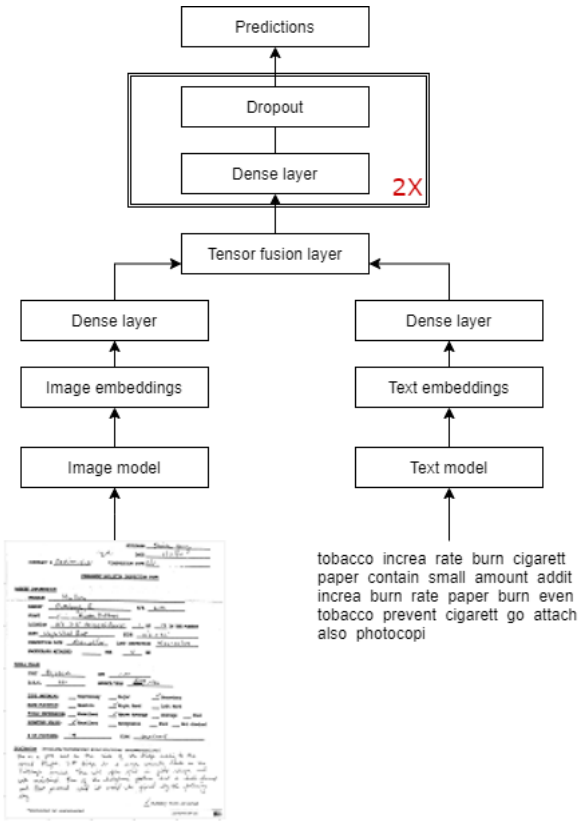


Figure 6.9: Tensor fusion architecture

Approach

The work of [Zadeh et al. (2017)] uses three dense layers after adding a custom layer for the computation of the outer product. In this work we propose to use two dense layers before the softmax layer and we perform a grid search to find suitable parameters for this approach. Computing the outer product across the whole dataset can be very expensive in terms of resources. Therefore, we adapt some of the work done in [Zadeh et al. (2017)] to create a custom tensor fusion layer which works on batches of the data. In the case of this work these are tensors of rank 2 with a dimension $[N, M]$ where N is the batch size and the M is the size of the embedding. To compute z^m we expand the 2D matrices into 3D cubes of size $[N, M, 1]$ and $[N, 1, M]$ and then compute the outer product (equation

6.1). After that we flatten the resulting 3D cube into a 2D matrix. The tensors produced by this approach tend to be large, that means that after flattening to 2D there are, in our case, 10000 features for each sample. This requires that we adapt our architecture to the large number of features. To address this, we propose to use two dense layers, the first of which acts as a filter for important features and the second as a layer where the multi modal representation is aggregated.

$$z^m = \begin{bmatrix} z^i \\ 1 \end{bmatrix} \bigotimes \begin{bmatrix} z^t \\ 1 \end{bmatrix} \tag{6.1}$$

6.4.4 Results

T3482 dataset

	Top 1 acc.	Top 3 acc.
Image model (fig 6.1)	**0.77558**	**0.9452**
Text model (fig 5.2)	0.70145	0.9109

	Late fusion	Late fusion (D)	Tensor fusion	Early fusion
Mean loss	0.698572	0.702117	0.796355	**0.605761**
Mean top 1 accuracy	0.795890	0.791297	0.793956	**0.820951**
Top 1 accuracy std	0.008909	0.009468	**0.002647**	0.003462
Mean top 3 accuracy	0.937550	0.938759	0.902297	**0.960395**
Top 3 accuracy std	0.004816	0.005442	0.008998	**0.003435**
MAX top 1 accuracy	0.811845	0.806608	0.796938	**0.824738**
MIN top 1 accuracy	0.783239	0.779613	0.789686	**0.815471**
Mean average recall	0.758894	0.756228	0.766272	**0.788132**
Mean average precision	0.828537	0.821711	0.834773	**0.852124**

Table 6.4: Multi modal fusion, 10 runs, balanced data split

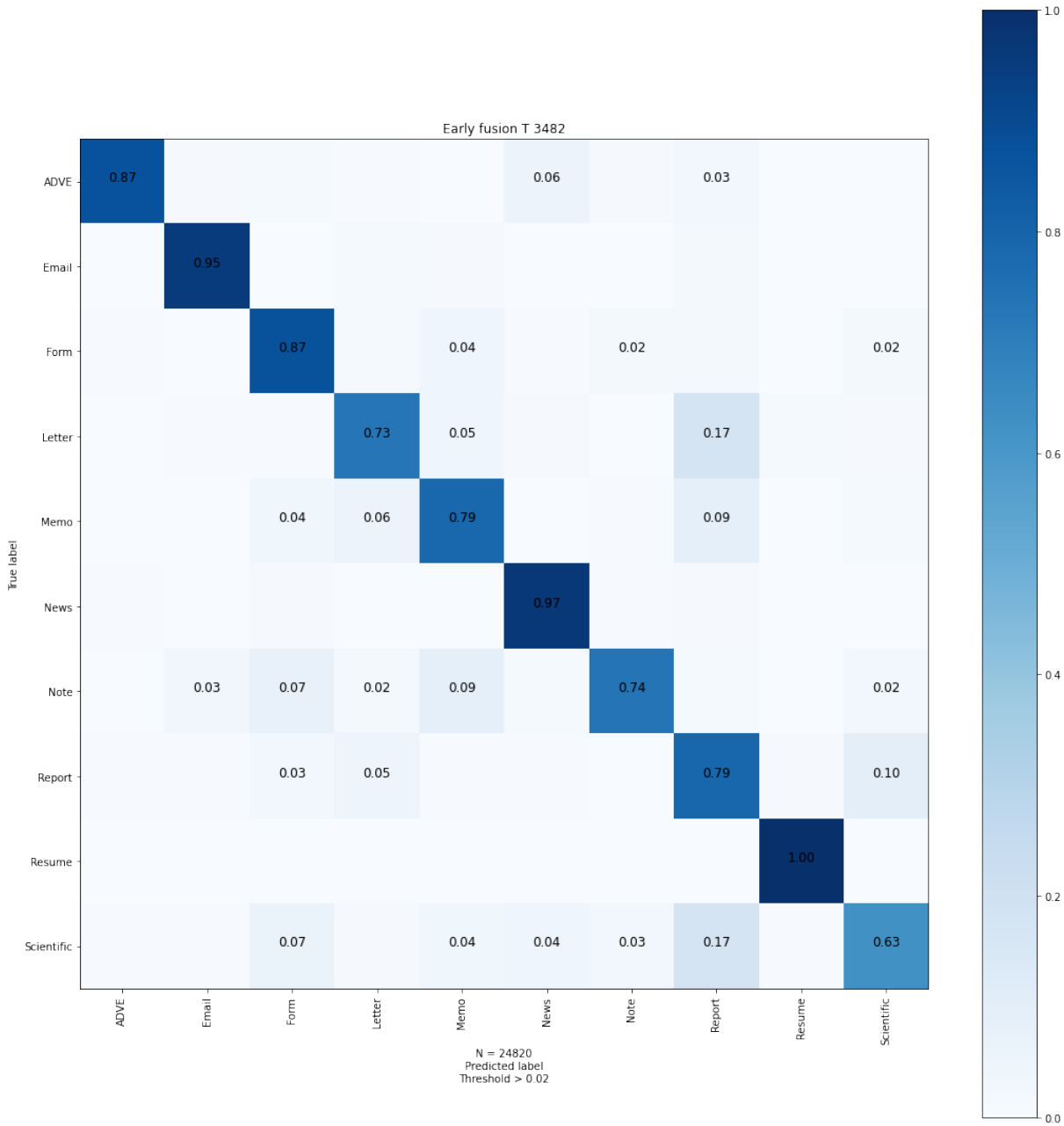


Figure 6.10: Early fusion confusion matrix for the dataset T3482, *balanced* baseline, 10 runs average

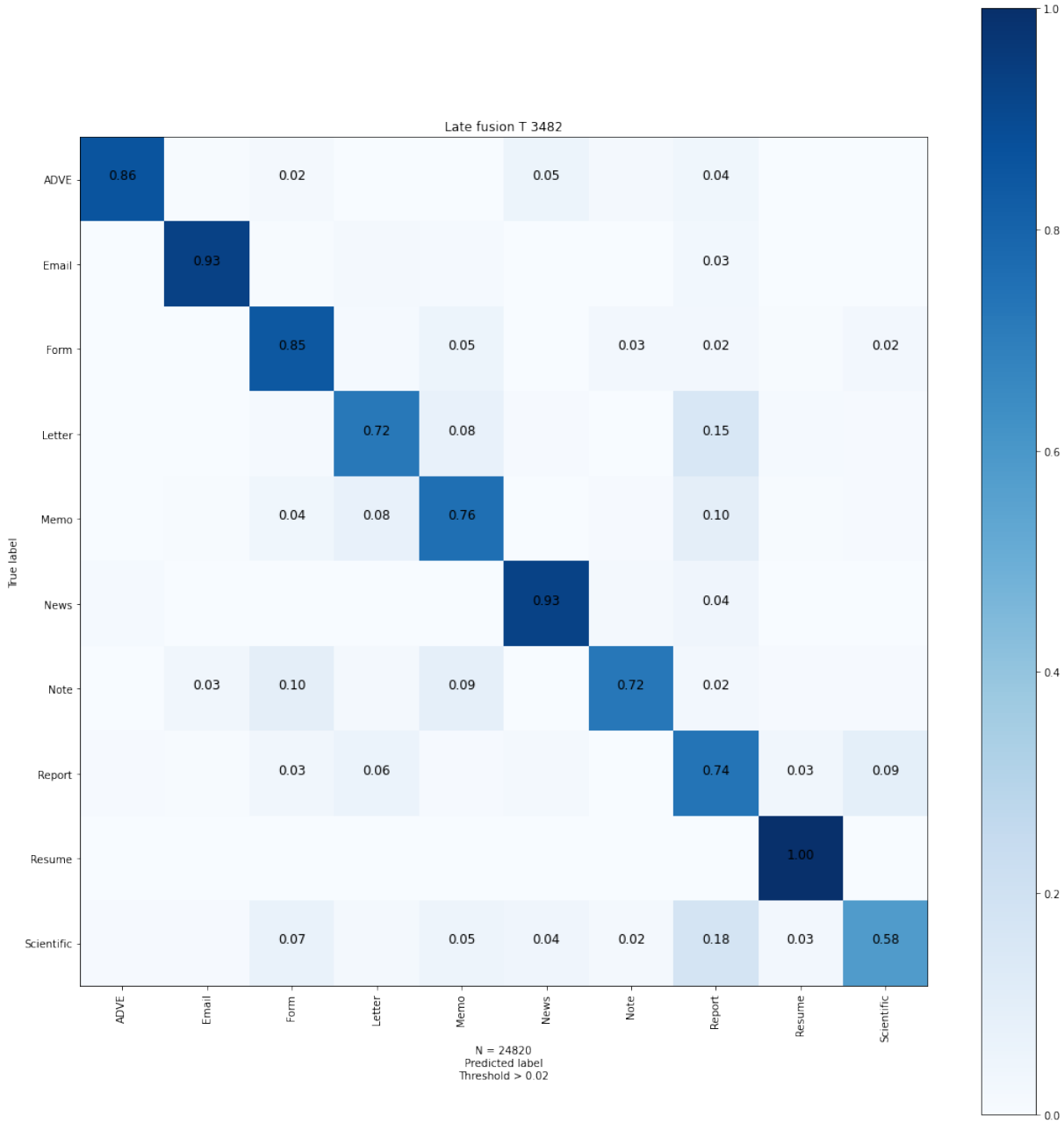


Figure 6.11: Late fusion confusion matrix - single dense layer NN, balanced split, 10 runs average

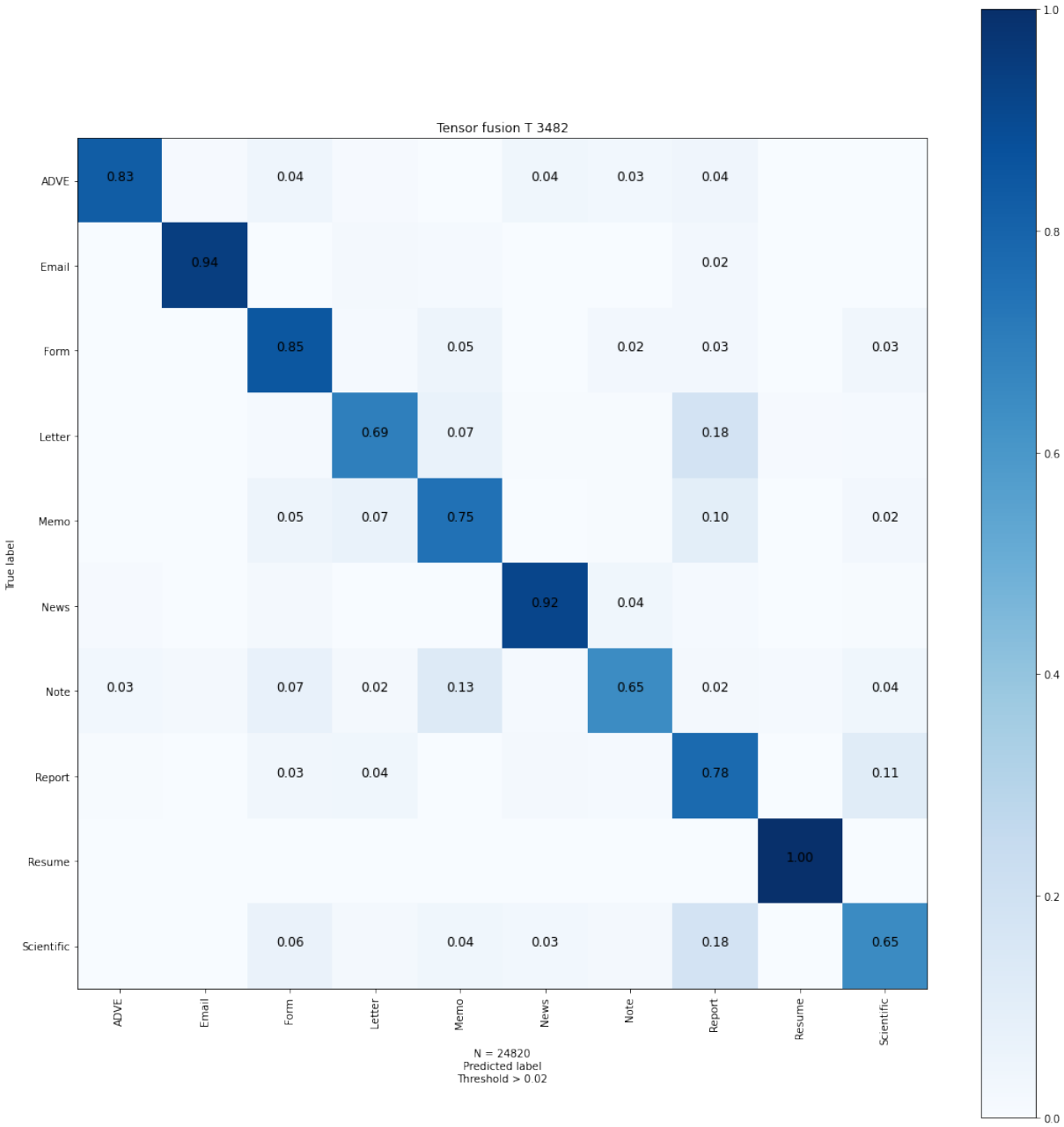


Figure 6.12: Tensor fusion confusion matrix, *balanced* baseline, 10 runs average

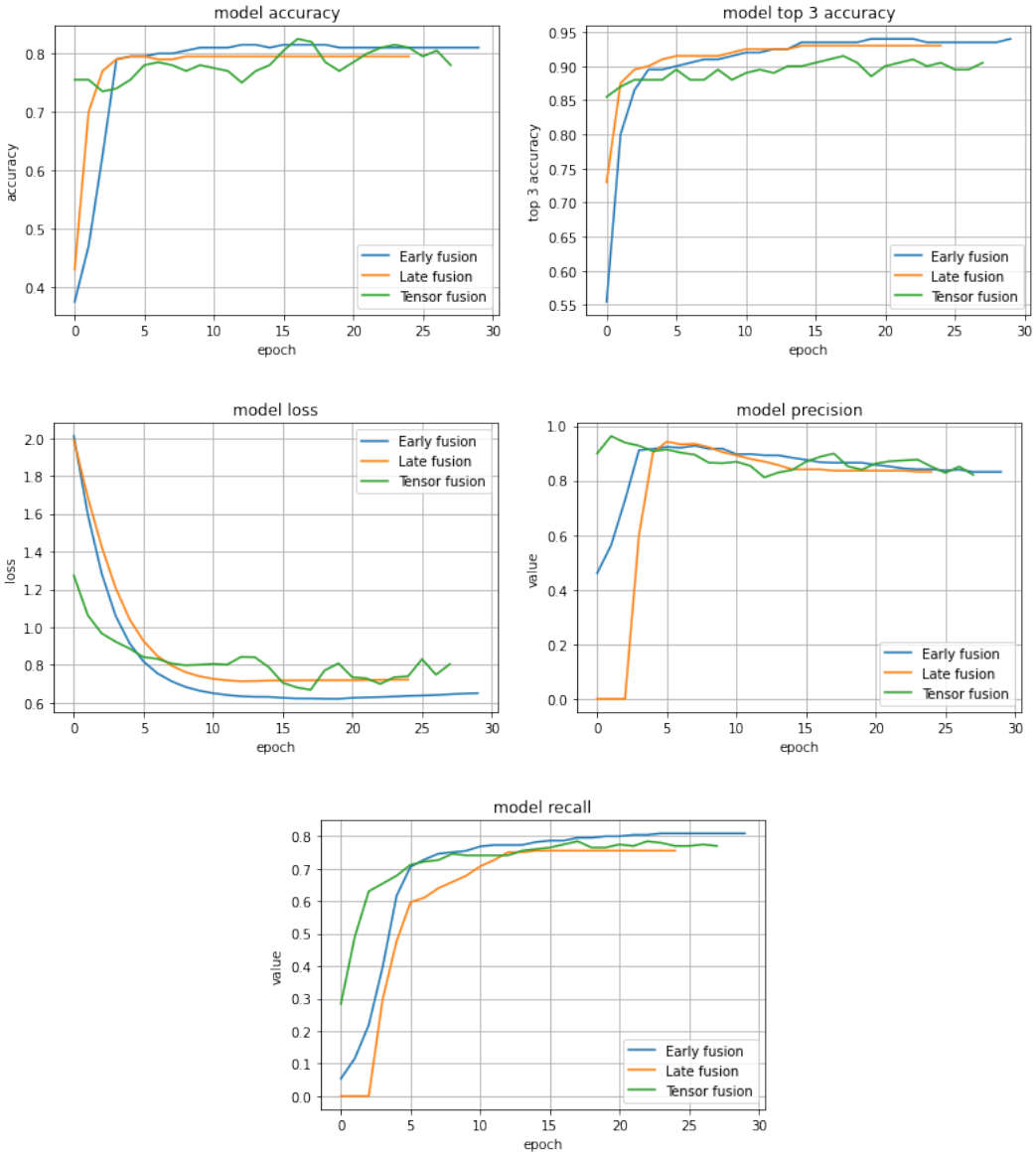


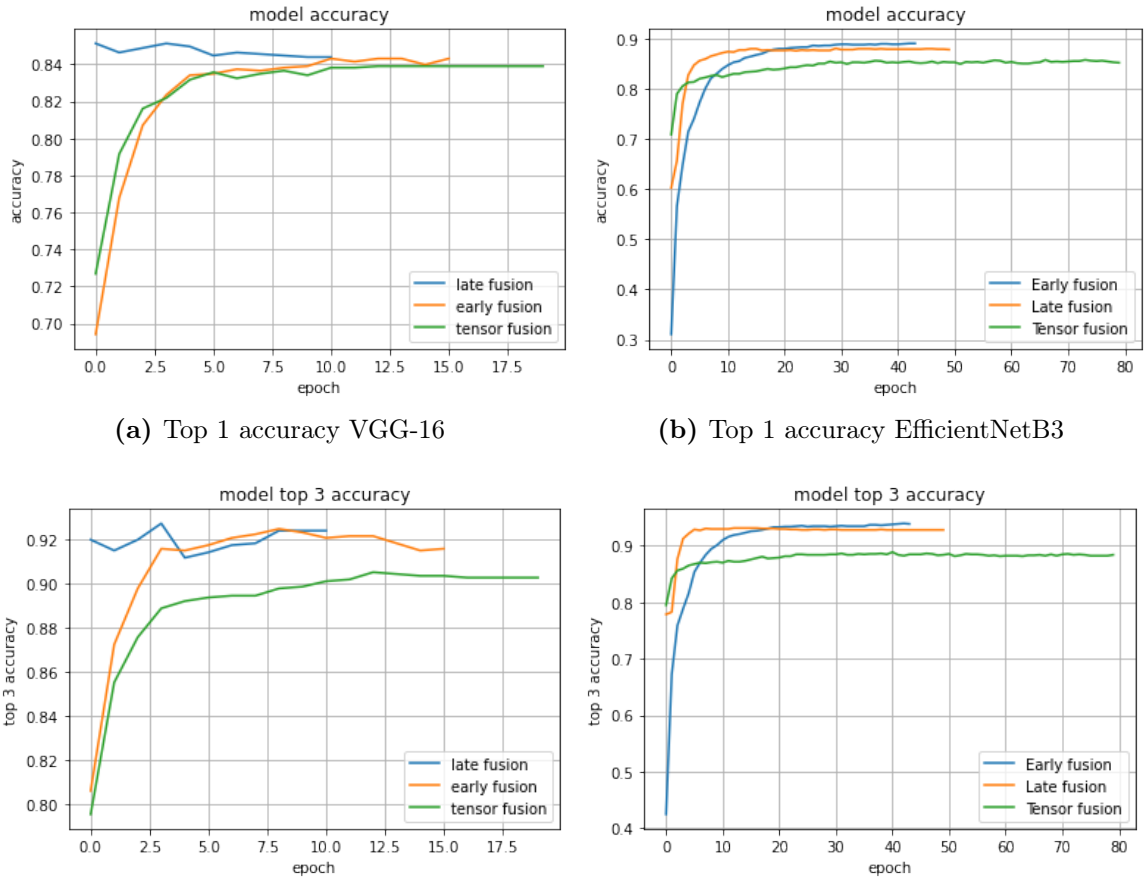
Figure 6.13: Validation measurements for training progress for each fusion approach for the *balanced* baseline on T3482. Single dense layer late fusion.

INTERNAL dataset

For the INTERNAL dataset we used the same configurations for early and late fusion as the one in T3482 as they seemed to work well. The tensor fusion configuration achieved worse results than the vision model. To address that we increased the sizes of the last two layers for the INTERNAL data set.

	Image model	Text model	Early fusion	Late fusion	Late fusion (D)	Tensor fusion
Top 1 Accuracy	0.8567	0.7797	**0.88416**	0.8759	0.8645	0.86125
Top 3 Accuracy	0.9368	0.8835	**0.9541**	0.9348	0.9397	0.8952
Mean recall	0.8497	0.76822	0.871	**0.872**	0.8593	0.8434
Mean precision	0.871	0.8127	**0.9068**	0.8817	0.8734	0.8964
F1 score	0.8596	0.7895	**0.8883**	0.8768	0.8662	0.8687

Table 6.5: Multi modal fusion results, all data split



(a) Top 1 accuracy VGG-16

(b) Top 1 accuracy EfficientNetB3

(c) Top 3 accuracy VGG-16 B3

(d) Top 3 accuracy EfficientNet B3

Figure 6.14: Validation measurements for training progress for each fusion approach for the *all data* data split on INTERNAL

7 Discussion

7.1 Image model

Our results for VGG-16 were not as good as those in [Afzal et al. (2017)] for the T3482 data set. This is perhaps due to not having access to the exact sizes of the dense layers and their corresponding dropout rates. Additionally, it could mean that the VGG-16 architecture does not work well with higher resolutions because we use a resolution of 300x300. Nonetheless, our first experiment shows that EfficientNet B3 achieved test set accuracy better than all base models reported in [Afzal et al. (2017)]. This is shown in table 6.2. The EfficientNet B3 architecture also benefits from faster training time as it converges faster than the VGG-16 architecture. For the INTERNAL dataset this speedup was more than 4 hours. Additionally, the gains on the INTERNAL dataset are larger in terms of accuracy by using this architecture. In conclusion, our experiments show that the architecture which uses VGG-16 (proposed in [Afzal et al. (2017)]) performs worse than EfficientNet B3 when the latter is used with a global max pooling layer (similar to the way [Audebert et al. (2020)] used the MobileNet architecture).

7.2 Text models

The work done in [Audebert et al. (2020)] clearly shows that better accuracy can be achieved with a 1D CNN model. This model is trained on the text by taking the first 500 letters of each document and ignoring the rest of the text. While this was attempted in this thesis, we found it to be too risky to apply it because in the same article the author specifies that the embeddings produced by fastText are far superior for words which are similar compared to other approaches. This can be seen in figure 7.1.

7.3 RandAugment

More consecutive operations with smaller magnitude affect the classifier trained on non-augmented data in a less negative way in comparison with fewer consecutive operations with a larger magnitude parameter. The images created through RandAugment should make our classifier less prone to mistakes whenever a given image is transformed for

Word pair	Similarities		
specifically	GloVe	ELMo	FastText
Specificalily	0.71	0.68	0.96
filter	GloVe	ELMo	FastText
fiilter	0.91	0.73	0.96
alcohol	GloVe	ELMo	FastText
Aleohol	0.40	0.69	0.88
Largely	GloVe	ELMo	FastText
Largly	0.25	0.81	0.98

(b) Word embeddings similarity for misspelled words.

Figure 7.1: Source - [Audebert et al. (2020)]

example by rotating it slightly whenever that image is included through a scanner. This is visible when the figures 6.5 and 6.6 are compared. While high values of both N (the number of transformations to apply consecutively) and M (the magnitude of each of the operations) seem to be less useful, it is harder to draw conclusions on what the optimal parameters are without checking the accuracy of a classifier for each possible combination. In this experiment we show that training only on a data set of images which are not augmented may hurt the performance of a classifier and that RandAugment can be used to augment the images to improve the robustness of classifiers.

7.4 Multi modal fusion

We view the results for T3482 as a test for how well a multi modal fusion can construct the actual representation. Our experiment shows that the ranking of the approaches on T3482 are close to the ones we achieved on the INTERNAL data set. In table 6.4 we can see that the early fusion and late fusion approaches were first and second respectively. This suggests that the parameters found on T3482 are a good fit for other document image classification tasks.

7.4.1 Early fusion

We found this method to be easy to implement and tune because it only has two parameters: the size of the dense layer after concatenation of the features and the dropout rate. Despite the simplicity of this method, it has shown the best performance on both the T3482 and

the INTERNAL datasets. This was surprising to us because [Åberg (2018)] achieved better results through the late fusion approach.

7.4.2 Late fusion

The number of patterns that late fusion can learn is limited by not allowing cross modal interactions directly from the representations of the individual modalities. This is perhaps the limiting factor when it comes to the smaller increases in accuracy when compared to early fusion. We also experimented with two layers after concatenation of the output probabilities but that was not as successful as the single dense layer configuration. Additionally, after consideration of the output probabilities of each modality, we noticed that often one of the probabilities tends to be high, often 0.9 or more. That is perhaps due to the model predicting the output probabilities for an image it was trained on. This means that the late fusion model will only learn which of the two classifiers to trust more. It appears that late fusion would be more useful when the individual classifiers are less sure about what the category is. This could be addressed by perhaps expanding the multi modal fusion training set by half the validation set or by using a smaller training set for both modalities. This could however lead to lower performance of the models for each modality and the multi modal fusion network.

7.4.3 Tensor fusion

Tensor fusion performed better than the best late fusion (with two dense layers) configuration we found for the T3482 dataset. This configuration, however, did not translate well to the INTERNAL dataset as it is visible in table 6.5. This is perhaps due to different datasets requiring different sizes for the embeddings of both images and text. We found that the search for suitable hyper parameter values for this approach can be far more difficult than for the other two approaches to fusion. That is because this approach has significantly more parameters - the number of dense layers, the size of the dense layers and dropout rates. Additionally, this approach appears to be more sensitive to the size of the embeddings. We attempted to use this approach with only a single layer with a ReLU activation, the same way we used it for early fusion but that did not yield sizable gains for the T3482 dataset. Additionally, we conducted a thorough search for parameter values for the two layer approach for INTERNAL described in figure 6.9 and finding suitable

parameter values was difficult due to the bigger size of the data set.

7.4.4 Usage of the model

The early fusion model performed best for both data sets. We propose the output of this approach to be used as suggestions rather than the top-1 predictions to be taken as they are. These suggestions would mean that the correct category is with a high (>.95%) probability within the top 3 categories. When compared to the performance of humans on the ImageNet data set which has been measured at 94.9% [Russakovsky et al. (2015)]. Despite not reaching that top-1 accuracy on the INTERNAL dataset, we can conclude that the early fusion approach top-3 accuracy of 95.41% (see table 6.5) should provide reliable predictions in real-life scenarios.

7.5 Hyper parameters

It is not clear what the size of the embeddings should be. In this work the size of the embeddings was chosen based on previous work and personal observations on what the best size may be. When it comes to comparison between the different methods multi modal fusion methods balancing the data set seems like a step in the right direction but a significant obstacle is that these models depend heavily on their hyper parameter values.

8 Further research

8.1 The size of images

The EfficientNet B3 network performs better than VGG-16 but training an even larger model might provide further increases in accuracy. We decided to leave that for further research due to the limited amount of memory available on the GPUs we have access to. The EfficientNet B7 architecture can be used for that. [Tan and Le (2019)] show improvements as the model scales up to B7, we suspect this will also be the case for document image classification. For this research we wanted to keep the results accessible to the general public but a future study may consider how a larger architecture performs. The experiments we performed in this work have not come without some choices which

may have decreased the performance of the model. The main ones were: what global pooling layer should be used for EfficientNet B3 (max or average) and what the sizes for the VGG-16 dense layers presented in [Afzal et al. (2017)] should be. Additionally, we initially attempted the experiments with a larger resolution and later found that the official implementation of EfficientNet includes a different resolution recommendation.

8.2 Transfer learning from RVL-CDIP

Training EfficientNet B3 on RVL-CDIP requires a lot more computational time than training it on T3482 or INTERNAL. This is the reason why we decided to leave it for a further study, however, the gains from having the pre-initialized weights trained on RVL-CDIP can be large. For example, in [Afzal et al. (2017)] the gains in average accuracy are above 10%. While such a gain happens because RVL-CDIP and T3482 are subsets of a common set, an improvement in accuracy for our own data set is expected as well.

8.3 Text model

The BERT model [Devlin et al. (2019)] has shown promising results, achieving state of the art accuracy on the RVL-CDIP dataset [Xu et al. (2019)]. This can help us improve the Text model results which lacks behind the results of the **Image** model for both of the data sets in this work. An interesting idea to explore is to match the OCR text into a pre-built subspace of vectors. This matching can be done by using Levenshtein distance. Perhaps a pre-built space can be created for fine grained tasks as the embeddings heavily depend on the context.

8.4 Tensor fusion

The tensor fusion approach can benefit from a more thorough investigation of suitable parameter values for larger data sets. Perhaps a sample of the data set can be used to determine what the best values are. Additionally, perhaps a different approach is needed to interpret the features from the tensor fusion layer. In conclusion, perhaps our configuration for the tensor fusion approach was sub optimal, and we should not exclude this approach as a potential improvement to early and late fusion. In figure 6.13 we can

see that the loss for this approach fluctuates more than the other two approaches. This perhaps means that two dense layers may be causing the model to overfit.

8.5 The effects of the data split

For the all data split we use 20% of the total data for the test set. Other work uses a different split - [Åberg (2018)] uses 72/8/20 split. It can be worth investigating how different approaches compare to each other. Additionally, training the multimodal fusion network is done on the training set. We found that to be the better configuration through running the algorithms multiple times on both configurations. However, it might be beneficial to train on half the validation data in addition to the training set data.

8.6 Attention mechanisms

We suspect that adding **inter** modal attention mechanisms can improve the performance of the tensor fusion model as it will help filter out irrelevant features. Tensor fusion appears to be prone to over fitting and being able to focus on certain features might improve both the performance and time to tune. An attempt to integrate attention mechanisms in an **intra**-modal fashion was done in [Åberg (2018)] but that was not more successful than the implementation-wise simpler late fusion approach.

8.7 Autoencoders

It can be interesting to explore how the representations constructed in the hidden layers of early and tensor fusion compares to the representations learned by an auto encoder.

8.8 RandAugment

We did not test whether RandAugment resulted in improvements in accuracy as it would increase the training set size significantly. Investigation into how to augment the images without adding a copy for each image in the training set might be an interesting area for further research.

9 Conclusions

9.1 Research questions

There are three questions that we asked in this thesis.

1. For the first question, we asked how does EfficientNet compare to state of art architectures? Is compound scaling applicable to the problem of document image classification?

   In this thesis, we performed a comparison between EfficientNet and VGG-16. In our first experiment we saw that applying compound scaling to an EfficientNet based architecture results in better performance than an architecture which uses VGG-16 on both the T3482 and INTERNAL data sets.

2. For our second question, we asked whether data augmentation should be applied to the document image classification problem and if RandAugment [Cubuk et al. (2020)] can make our classifier work better with images that are transformed (for example, rotated or skewed).

   In our second experiment, we showed that the performance of the EfficientNet model is better when data augmentation is used on the training set. In conclusion, data augmentation should be used for real-life scenarios. While RandAugment can help with improving the robustness of EfficientNet, the parameter values should be chosen carefully.

3. Our third question was: How do tensor fusion methods [Zadeh et al. (2017)] compare to the more well known methods of early and late fusion on the balanced data split for T3482? Does the model configuration found do well on INTERNAL? Does the best method we find on the balanced data set split perform the best on the all data split?

   In our third experiment, tensor fusion performed worse than early fusion and has similar performance as late fusion. The configuration found on the balanced split of T3482 did not translate well for tensor fusion to the all data split on the INTERNAL data set. Additionally, the tuning process for tensor fusion was more difficult due

to the number of hyper parameters. This may have caused the classifier to under perform. The best performing method found on the balanced split for T3482 was also the best performing method for the INTERNAL dataset.

9.2 Conclusion

Predicting document image categories is a problem that has generated interest in both research and commercial sense. This interest is sparked by the vast number of digital documents that companies and institutions need to manage. There is consensus in the scientific community about the use of machine learning models to tackle this problem. The architecture used in this work, based on the EfficientNet B3 architecture introduced in [Tan and Le (2019)], has achieved large improvements in accuracy in both datasets used in this work (compared to VGG-16). Multi modal fusion approaches can effectively combine machine learning models in order to tackle the problem but significant challenges remain. We experimented with the tensor fusion approach and our observations suggest that there is likely no one parameter configuration suitable for all data sets. The differences when these documents contain, for example, more shapes (in the INTERNAL dataset) or more text (in the T3482 dataset) are perhaps too large. There are also other factors to take into account, such as, the number of OOV words and the size of the embeddings. Additionally, despite our best efforts on tuning, our observations show that the hyper parameter values for the tensor fusion approach [Zadeh et al. (2017)] we chose in this thesis do not generalize well across datasets. This could perhaps be tackled by inter-modal attention mechanisms to balance the contribution of the different modalities or by changing the number of dense layers and their sizes. The tensor fusion approach did yield improvements over our best configuration for late fusion (with two layers) but the results are still below the early fusion and late fusion with single layer approaches. Tensor fusion also has more parameters to be adjusted - in particular, the number of dense layers, their size and the dropout rate that can accompany them.

In conclusion, the best result on a real life data set gathered at Radial SG was achieved though early fusion with an F1 score of 0.8883 and a 88.41% and 95.41%, top 1 and top 3 accuracy respectively.

References

Åberg, L. (2018). Multimodal classification of second-hand e-commerce ads (dissertation). retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-233324.

Afzal, M. Z., Capobianco, S., Malik, M. I., Marinai, S., Breuel, T. M., Dengel, A., and Liwicki, M. (2015). Deepdocclassifier: Document classification with deep convolutional neural network. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1111–1115.

Afzal, M. Z., Kolsch, A., Ahmed, S., and Liwicki, M. (2017). Cutting the error by half: Investigation of very deep cnn and advanced training strategies for document image classification. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*.

Audebert, N., Herold, C., Slimani, K., and Vidal, C. (2020). Multimodal deep networks for text and image-based document classification. In Cellier, P. and Driessens, K., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 427–443, Cham. Springer International Publishing.

Bai, X., Yang, M., Lyu, P., Xu, Y., and Luo, J. (2018). Integrating scene text and visual appearance for fine-grained image classification. *IEEE Access*, 6:66322–66335.

Baltrusaitis, T., Ahuja, C., and Morency, L.-P. (2017). Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359. Similarity Matching in Computer Vision and Multimedia.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Chen, N. and Blostein, D. (2007). A survey of document image classification: problem statement, classifier architecture and performance evaluation. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(1):1–16.

Cubuk, E. D., Zoph, B., Mané, D., Vasudevan, V., and Le, Q. V. (2019). Autoaugment: Learning augmentation strategies from data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Das, A., Roy, S., Bhattacharya, U., and Parui, S. K. (2018). Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3180–3185.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Ebersbach, M., Herms, R., and Eibl, M. (2017). Fusion methods for icd10 code classification of death certificates in multilingual corpora. In *CLEF*.

Engin, D., Emekligil, E., Oral, B., Arslan, S., and Akpınar, M. (2019). Multimodal deep neural networks for banking document classification. retrieved from https://www.researchgate.net/publication/336552928.

Guo, W., Wang, J., and Wang, S. (2019). Deep multimodal representation learning: A survey. *IEEE Access*, 7:63373–63394.

Harley, A. W., Ufkes, A., and Derpanis, K. G. (2015). Evaluation of deep convolutional nets for document image classification and retrieval. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 991–995.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 1026–1034, USA. IEEE Computer Society.

Hertel, L., Barth, E., Kaster, T., and Martinetz, T. (2015). Deep convolutional neural networks as generic feature extractors. *2015 International Joint Conference on Neural Networks (IJCNN)*.

Jin, H., Song, Q., and Hu, X. (2019). Auto-keras: An efficient neural architecture search system. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*.

Kang, L., Kumar, J., Ye, P., Li, Y., and Doermann, D. (2014). Convolutional neural networks for document image classification. In *2014 22nd International Conference on Pattern Recognition*, pages 3168–3172.

Kay, A. (2007). Tesseract: An open-source optical character recognition engine. *Linux J.*, 2007(159):2.

Kenter, T., Borisov, A., and de Rijke, M. (2016a). Siamese cbow: Optimizing word embeddings for sentence representations. *ArXiv*, abs/1606.04640.

Kenter, T., Borisov, A., and de Rijke, M. (2016b). Siamese CBOW: Optimizing word embeddings for sentence representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 941–951, Berlin, Germany. Association for Computational Linguistics.

Kermany, D., Goldbaum, M., Cai, W., Valentim, C., Liang, H.-Y., Baxter, S., McKeown, A., Yang, G., Wu, X., Yan, F., Dong, J., Prasadha, M., Pei, J., Ting, M., Zhu, J., Li, C., Hewett, S., Dong, J., Ziyar, I., and Zhang, K. (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172:1122–1131.e9.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2017). On

large-batch training for deep learning: Generalization gap and sharp minima. *ArXiv*, abs/1609.04836.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kornblith, S., Shlens, J., and Le, Q. V. (2019). Do better imagenet models transfer better? *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2656–2666.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861 – 867.

Liu, Z., Shen, Y., Lakshminarasimhan, V. B., Liang, P. P., Bagher Zadeh, A., and Morency, L.-P. (2018). Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256, Melbourne, Australia. Association for Computational Linguistics.

Long, X., Gan, C., de Melo, G., Liu, X., Li, Y., Li, F., and Wen, S. (2018). Multimodal keyless attention fusion for video classification. In *AAAI*.

Nagy, G. (2000). Twenty years of document image analysis in pami. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62.

Pondenkandath, V., Seuret, M., Ingold, R., Afzal, M. Z., and Liwicki, M. (2017). Exploiting state-of-the-art deep learning methods for document image analysis. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 05, pages 30–35.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Sato, J. and Cipolla, R. (1999). Affine reconstruction of curved surfaces from uncalibrated views of apparent contours. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 25(11):1188–1197.

Snoek, C., Worring, M., and Smeulders, A. (2005). Early versus late fusion in semantic video analysis. *Proceedings of ACM Multimedia*, pages 399–402.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1931.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *ICANN*.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. cite arxiv:1905.11946Comment: Published in ICML 2019.

Tensmeyer, C. and Martinez, T. (2017). Analysis of Convolutional Neural Networks for Document Image Classification. *arXiv e-prints*, page arXiv:1708.03273.

Vielzeuf, V., Lechervy, A., Pateux, S., and Jurie, F. (2019). Centralnet: A multilayer approach for multimodal fusion. In Leal-Taixé, L. and Roth, S., editors, *Computer Vision – ECCV 2018 Workshops*, pages 575–589, Cham. Springer International Publishing.

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5(2):241 – 259.

Xie, Q., Hovy, E. H., Luong, M.-T., and Le, Q. V. (2019). Self-training with noisy student improves imagenet classification. *ArXiv*, abs/1911.04252.

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 2048–2057. JMLR.org.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., and Zhou, M. (2019). Layoutlm: Pre-training of text and layout for document image understanding. *ArXiv*, abs/1912.13318.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems (NIPS)*, 27.

Zadeh, A., Chen, M., Poria, S., Cambria, E., and Morency, L.-P. (2017). Tensor fusion network for multimodal sentiment analysis. *ArXiv*, abs/1707.07250.

Zoph, B. and Le, Q. V. (2017). Neural architecture search with reinforcement learning. *ArXiv*, abs/1611.01578.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710.

Appendix

```
# Preprocess text
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
def preprocess(s):
    lemmatizer = WordNetLemmatizer( )
    letters_only = re.sub("[^\w\s]", " ", s)
    s = lemmatizer.lemmatize(s)
    words = word_tokenize(letters_only.lower())
    stops = set(stopwords.words("english"))
    meaningful_words = [ stemmer.stem(re.sub('[^a-zA-Z0-9 \n\.]', '', w)) for w
        in words if not w in stops ]
    final = ' '.join([w for w in meaningful_words])
    return final
```

```
# Hyper parameters used to train the text model
'dense_layer_1_size' : [2048],
'dropout_rate_dense_layer_1': [0.2],
'dense_layer_2_size' : [2048],
'dropout_rate_dense_layer_1': [0.2],
'dense_layer_3_size' : [100],
'dropout_rate_dense_layer_1': [0.2],
'optimizer' : ['Adam'],
'learn_rate': [0.001],
'batch_size' : [128],
'momentum' : [0.9],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
```

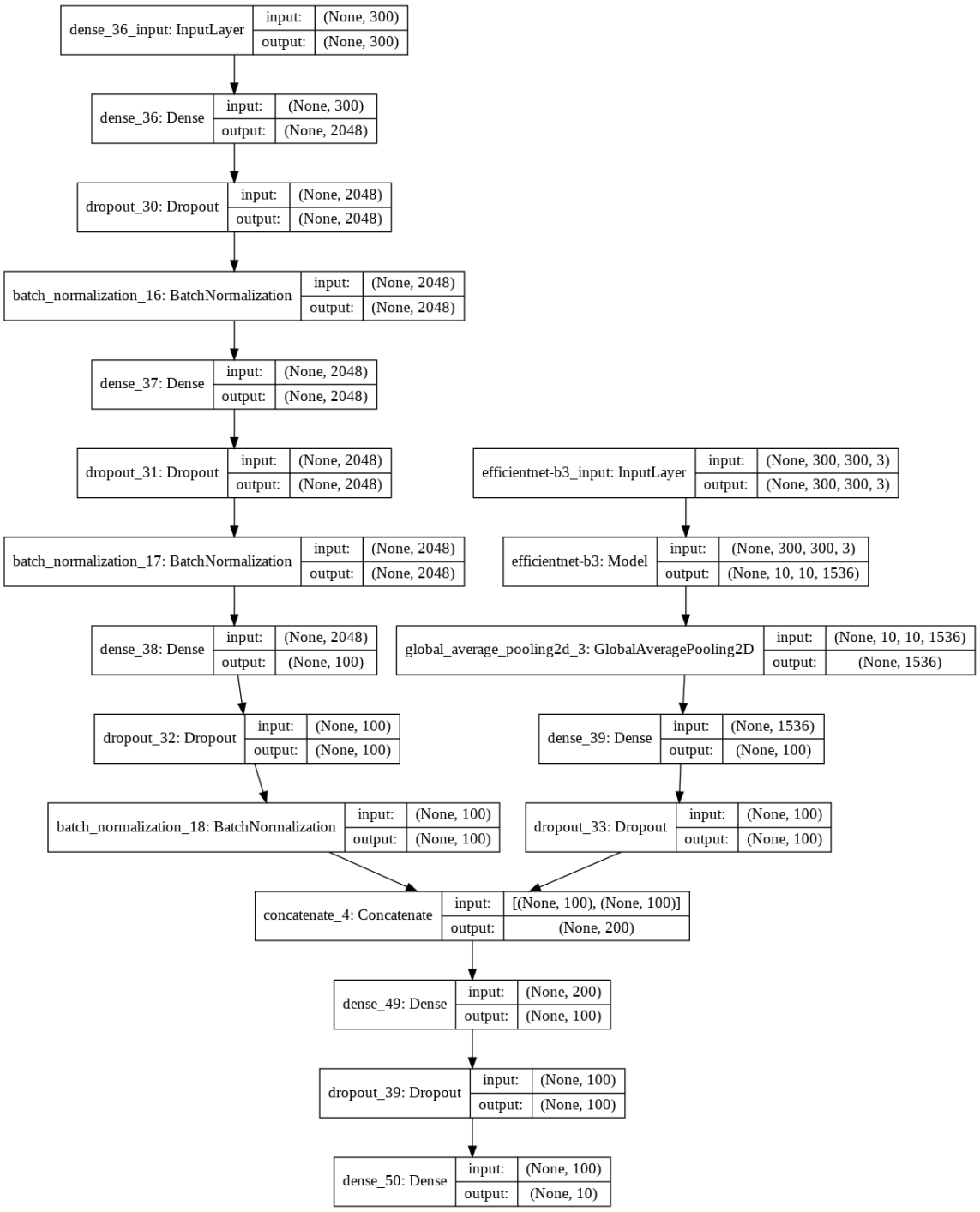


Figure A0.1: Early fusion architecture

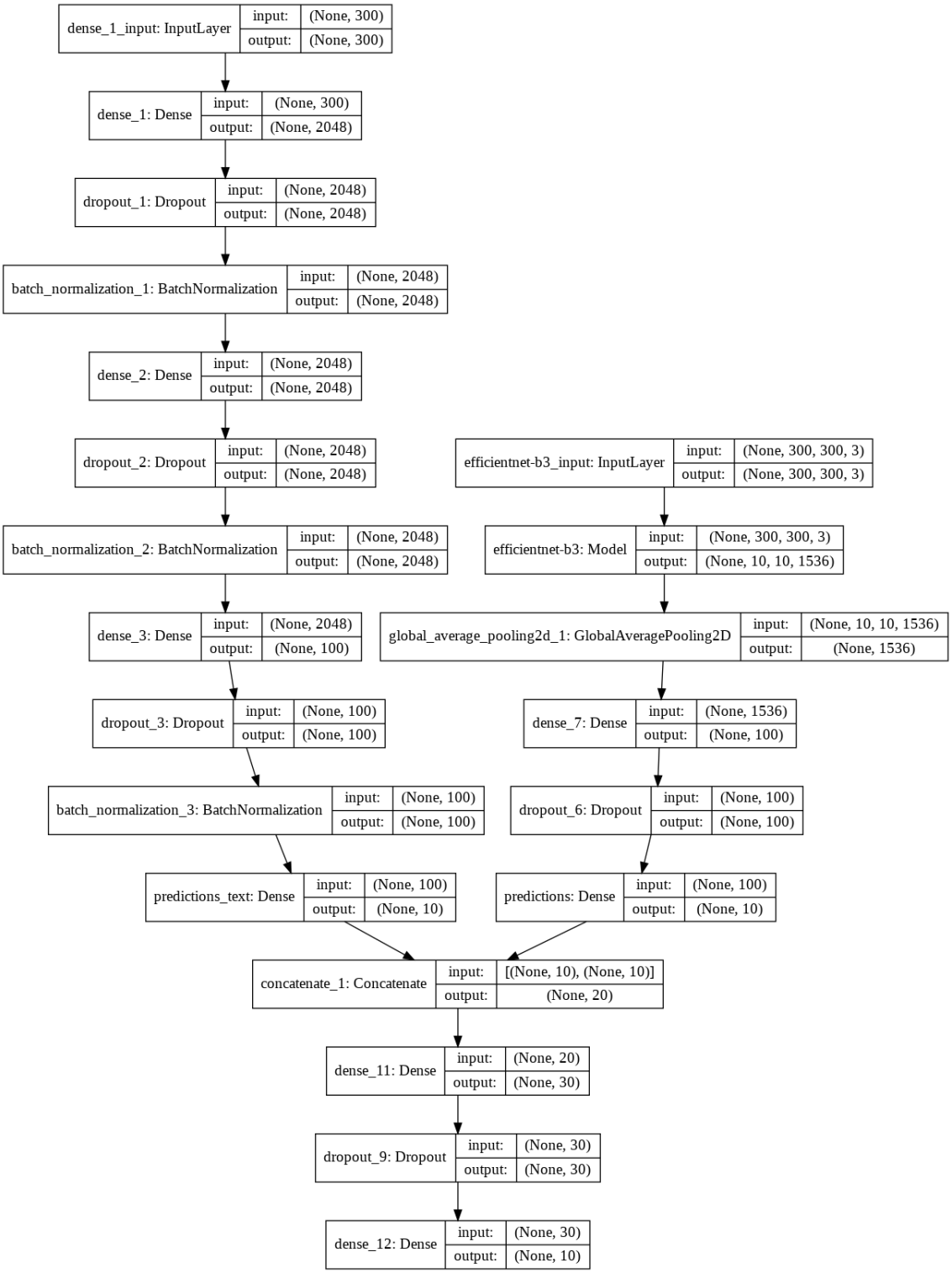


Figure A0.2: Late fusion architecture

Figure A0.3: Tensor fusion architecture, the lambda layer computes the outer product of the input tensors and flattens the 3D cube to a 2D matrix

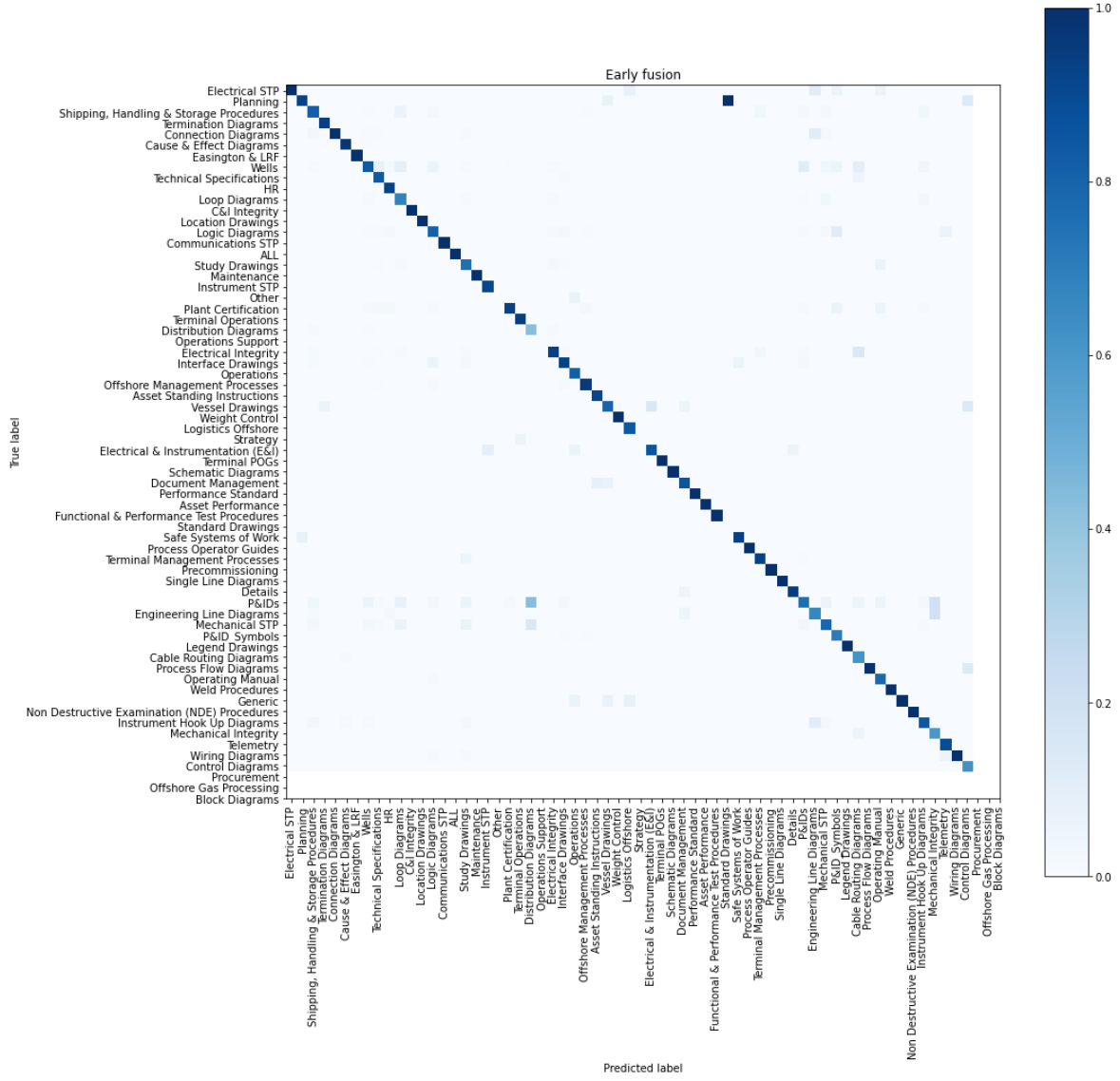


Figure A0.4: Early fusion confusion matrix. INTERNAL dataset

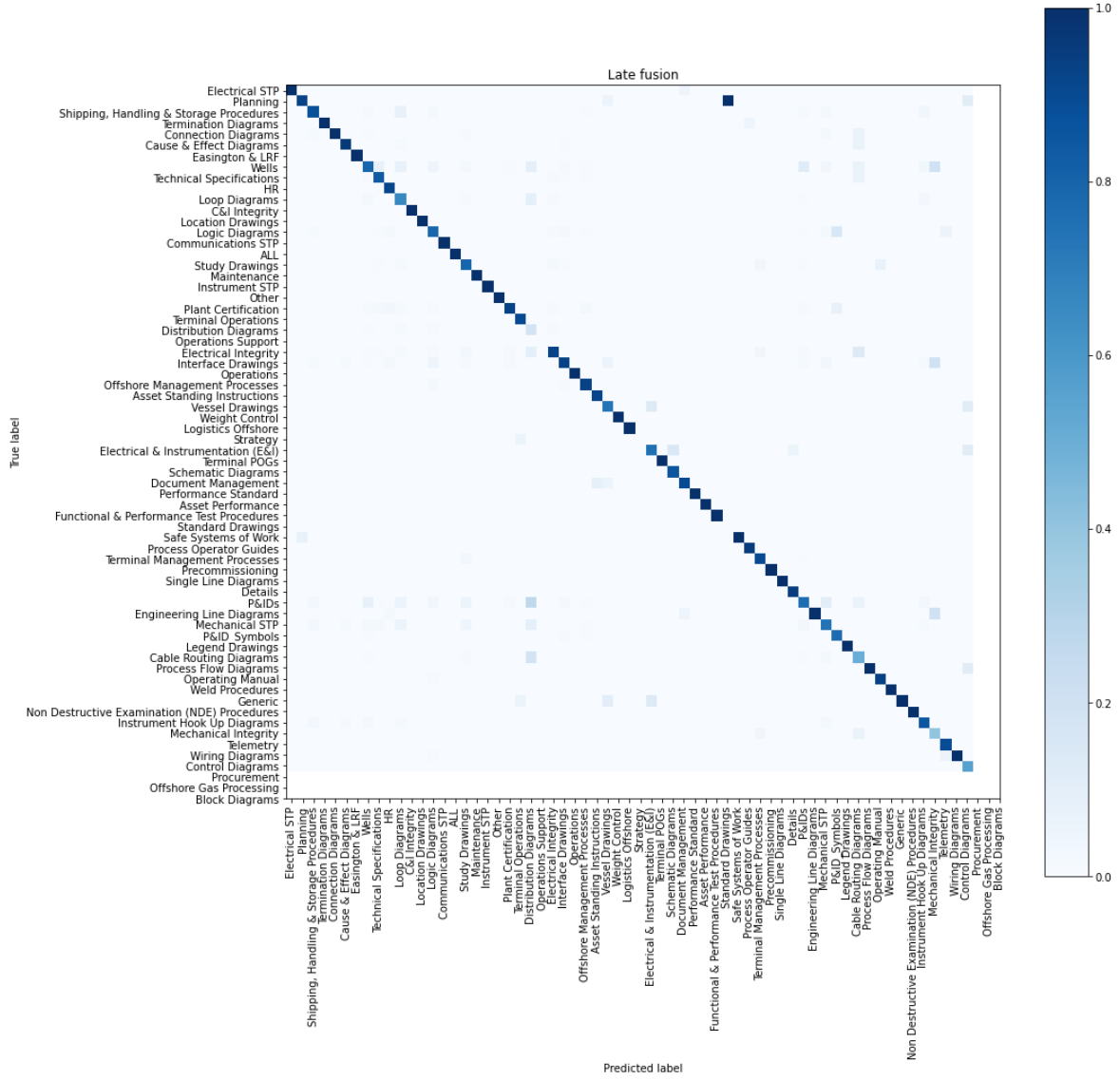


Figure A0.5: Late fusion confusion matrix. INTERNAL dataset

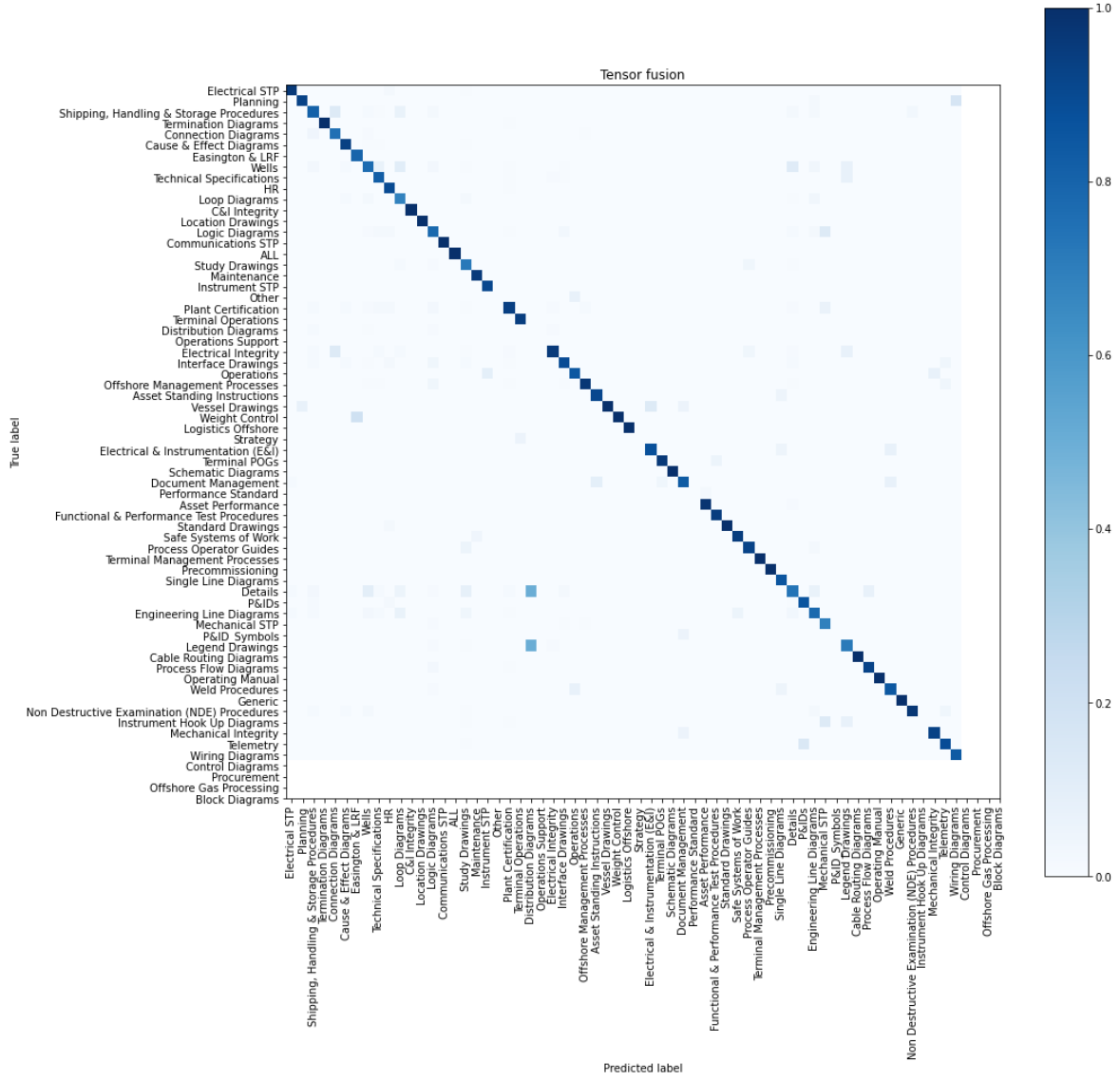


Figure A0.6: Tensor fusion confusion matrix. INTERNAL dataset

```
'early_stopping_patience' : [10],

Reduction of learning rate on plateau: 10% and factor=0.1, patience=4,
    verbose=1, epsilon=1e-4, mode='max',
```

```
# Grid search parameters used to train the image model - EfficientNet networks
'dense_layer_size' : [100],
'dropout_rate': [0.5],
'optimizer' : ['Adam'],
'learn_rate': [0.0001],
'batch_size' : [12],
'final_dense_layer_regularization' : [kernel_regularizer=regularizers.l2(0.001),
                                      activity_regularizer=regularizers.l1(0.001)],
Reduction of learning rate on plateau: 10% and factor=0.1, patience=4,
    verbose=1, epsilon=1e-4, mode='max'
```

```
# Hyper parameters used to train the image model - EfficientNet networks
'dense_layer_size' : [100],
'dropout_rate': [0.5],
'optimizer' : ['Adam'],
'learn_rate': [0.0001],
'batch_size' : [12],
'final_dense_layer_regularization' : [kernel_regularizer=regularizers.l2(0.001),
                                      activity_regularizer=regularizers.l1(0.001)],
Reduction of learning rate on plateau: 10% and factor=0.1, patience=4,
    verbose=1, epsilon=1e-4, mode='max'
```

```
# Hyper parameters used to train the image model - VGG network
'dense_layer_size' : [100],
'dropout_rate': [0.5],
```

```
'optimizer' : ['Adam'],
'learn_rate': [0.00001],
'batch_size' : [12],
Reduction of learning rate on plateau: 10% and factor=0.1, patience=4,
    verbose=1, epsilon=1e-4, mode='max'
```

```
# Grid search parameters used to train the late fusion model on T3482 for NN
    model
'dense_layer_size' : [30, 50, 80, 110, 140, 170, 200, 250],
'dropout_rate': [0.2, 0.3, 0.5, 0.6, 0.7 ,0.8],
'optimizer' : ['Adam'],
'learn_rate': [0.1],
'batch_size' : [128],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [5]
```

```
# Optimal parameters used to train the late fusion model on T3482 for single
    layer NN model, early stopping after epoch 12.
'dense_layer_size' : [30],
'dropout_rate': [0.6],
'optimizer' : ['Adam'],
'learn_rate': [0.001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_acc',
'early_stopping_patience_policy' : 'max',
'pearly_stopping_patience' : [5]
```

```
# Searched parameters used to train the late fusion model on T3482 for double
    layer NN model
'dense_layer_1_size' : [30, 40, 50, 60, 70, 80, 90],
'dense_layer_2_size' : [30, 40, 50, 60, 70, 80, 90],
'dropout_rate': [0.5],
'optimizer' : ['Adam'],
'learn_rate': [0.001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'early_stopping_patience' : [5],
```

```
# Found optimal parameters used to train the late fusion model on T3482 for
    double layer NN model
'dense_layer_1_size' : [30],
'dense_layer_2_size' : [30],
'dropout_rate': [0.5],
'optimizer' : ['Adam'],
'learn_rate': [0.001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_acc',
'early_stopping_patience_policy' : 'min',
'early_stopping_patience' : [5],
```

```
# Grid search parameters used to train the early fusion model on T3482 for
    single layer NN model
'dense_layer_size' : [150, 170, 190, 200, 220, 240],
'dropout_rate': [0.6, 0.65, 0.7],
'optimizer' : ['Adam'],
```

```
'learn_rate': [0.0001],
'batch_size' : [128],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [5]
```

```
# Optimal parameters used to train the early fusion model on T3482 for single
    layer NN model
'dense_layer_size' : [200],
'dropout_rate': [0.6],
'optimizer' : ['Adam'],
'learn_rate': [0.0001],
'batch_size' : [128],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [5]
```

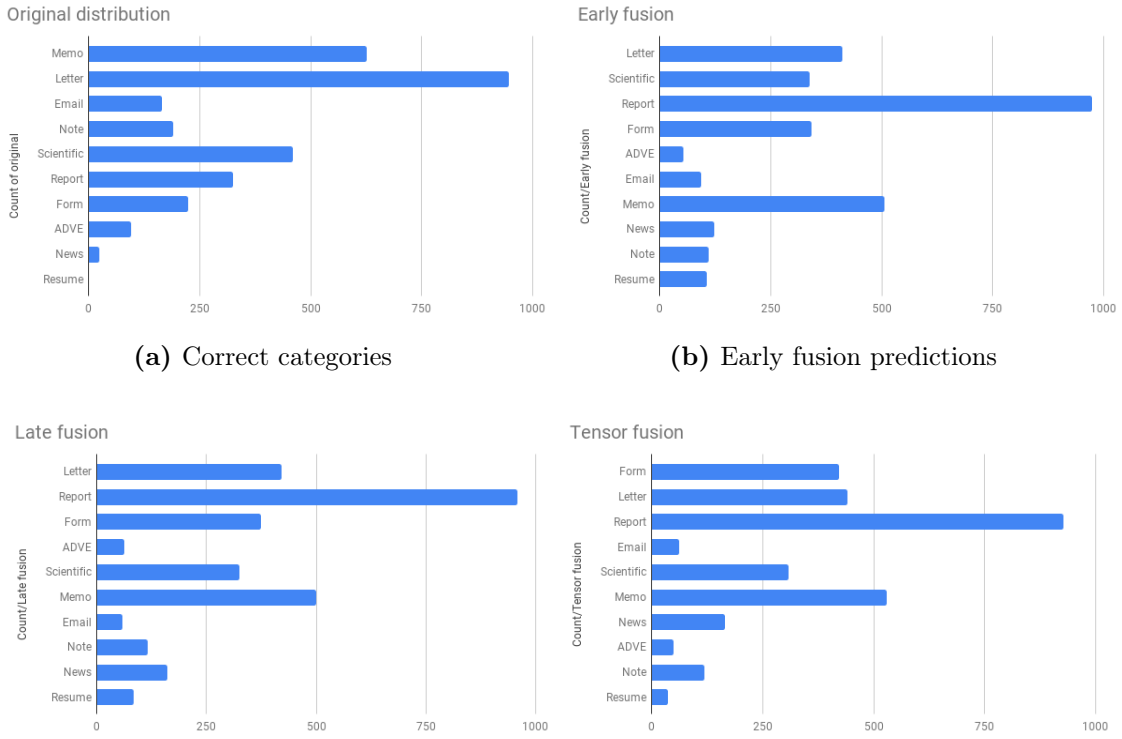
```
# Grid search parameters used to train the tensor fusion model on T3482
'post_fusion_layer_1_size' : [70, 80, 90, 100, 110, 120, 130, 200 , 300],
'post_fusion_layer_2_size' : [70, 80, 90, 100, 110, 120, 130],
'post_fusion_layer_1_activation' : 'relu',
'optimizer' : ['Adam'],
'text_embedding_size' : [300],
'dropout_rate': [0.6, 0.63, 0.7],
'learn_rate': [0.0001, 0.00001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
```

```
'pearly_stopping_patience' : [15]
```

```
# Optimal parameters used to train the tensor fusion model on T3482
'post_fusion_layer_1_size' : [120],
'post_fusion_layer_2_size' : [70],
'post_fusion_layer_1_activation' : 'relu',
'post_fusion_layer_2_activation' : 'sigmoid',
'text_embedding_size' : [300],
'dropout_rate': [0.6],
'optimizer' : ['Adam'],
'learn_rate': [0.0001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [20]
```

```
# Grid search parameters used to train the tensor fusion model on INTERNAL
'post_fusion_layer_1_size' : [70, 90, 110, 130, 200, 250, 300],
'post_fusion_layer_2_size' : [70, 80, 90, 100, 110],
'post_fusion_layer_1_activation' : 'relu',
'optimizer' : ['Adam'],
'text_embedding_size' : [300],
'dropout_rate': [0.6, 0.63, 0.7],
'learn_rate': [0.0001, 0.00001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [15]
```

```
# Optimal parameters used to train the tensor fusion model on T3482
'post_fusion_layer_1_size' : [200],
'post_fusion_layer_2_size' : [70],
'post_fusion_layer_1_activation' : 'relu',
'post_fusion_layer_2_activation' : 'sigmoid',
'text_embedding_size' : [300],
'dropout_rate': [0.6],
'optimizer' : ['Adam'],
'learn_rate': [0.0001],
'batch_size' : [16],
'epochs' : [1000],
'early_stopping_patience_parameter' : 'val_loss',
'early_stopping_patience_policy' : 'min',
'pearly_stopping_patience' : [20]
```



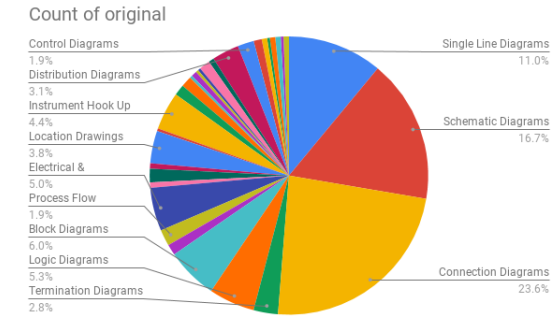
(a) Correct categories

(b) Early fusion predictions

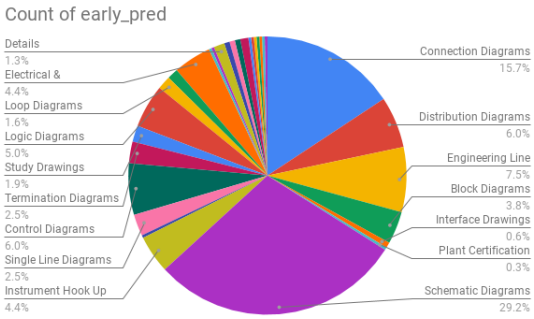
(c) Late fusion predictions

(d) Tensor fusion fusion predictions

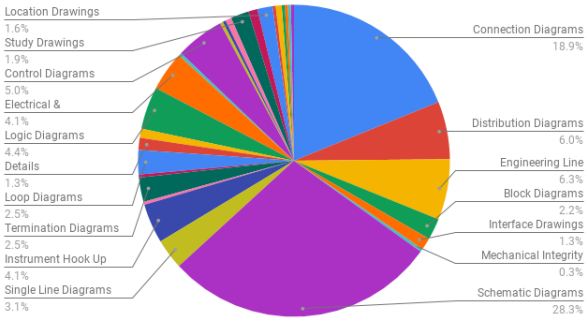
Figure A0.7: Wrongly predicted items of 10 runs for each of the fusion methods for T3482. Single dense layer late fusion. N = 3060



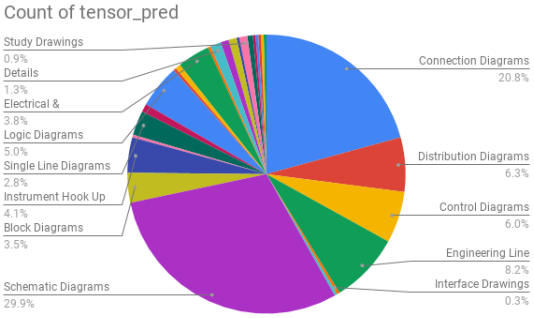
(a) Correct categories



(b) Early fusion predictions



(c) Late fusion predictions



(d) Tensor fusion fusion predictions

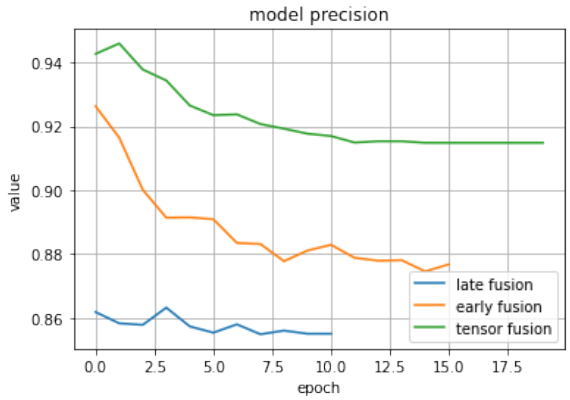
Figure A0.8: Wrongly predicted items for each of the fusion methods for INTERNAL

FastText parameters (default to date, taken from the official website[7])

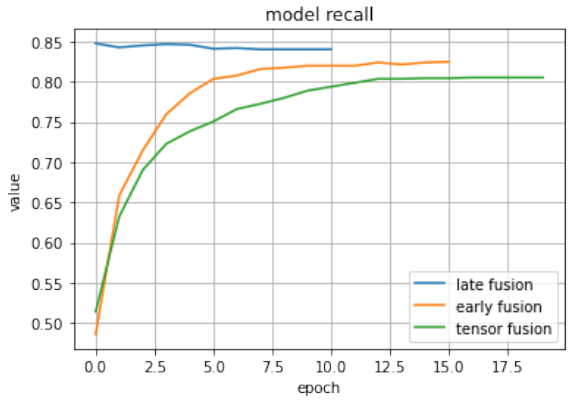
```
The following arguments for the dictionary are optional:
-minCount         minimal number of word occurrences [1]
-minCountLabel    minimal number of label occurrences [0]
-wordNgrams       max length of word ngram [1]
-bucket           number of buckets [2000000]
-minn             min length of char ngram [0]
-maxn             max length of char ngram [0]
-t                sampling threshold [0.0001]
-label            labels prefix [__label__]

The following arguments for training are optional:
-lr               learning rate [0.1]
-lrUpdateRate     change the rate of updates for the learning rate [100]
-dim              size of word vectors [100]
```

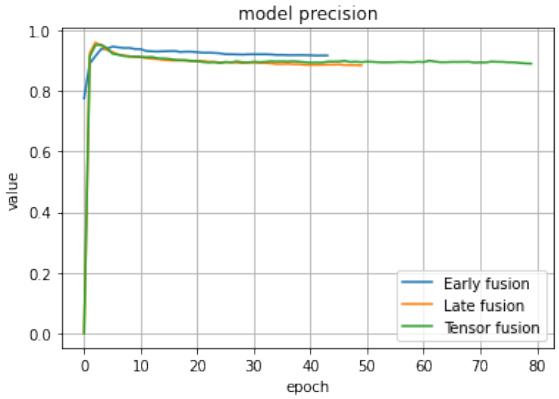
[7] https://fasttext.cc/docs/en/options.html



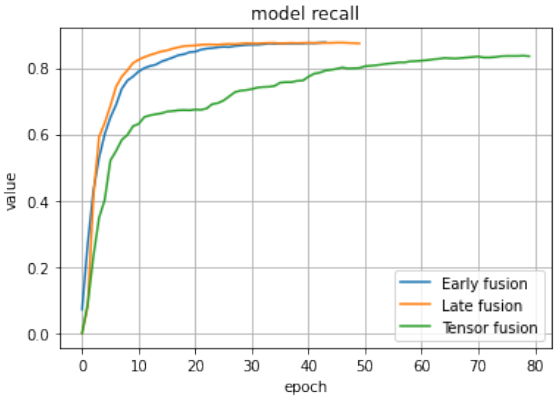
(a) Average precision VGG-16 B3



(b) Average recall VGG-16



(c) Average precision EfficientNet



(d) Average recall EfficientNet

Figure A0.9: Validation measurements for training progress for each fusion approach for the *all data* data split on INTERNAL

```
-ws                 size of the context window [5]
-epoch              number of epochs [5]
-neg                number of negatives sampled [5]
-loss               loss function {ns, hs, softmax} [softmax]
-thread             number of threads [12]
-pretrainedVectors pretrained word vectors for supervised learning []
-saveOutput         whether output params should be saved [0]
```

```
# Packages and their versions
absl-py                   0.9.0
anaconda-client           1.7.2
anaconda-navigator        1.9.12
```

```
asn1crypto                  1.3.0
astor                       0.8.1
attrs                       19.3.0
backcall                    0.1.0
backports.functools-lru-cache 1.6.1
backports.tempfile          1.0
backports.weakref           1.0.post1
beautifulsoup4              4.8.2
bleach                      3.1.0
blinker                     1.4
boto                        2.49.0
boto3                       1.12.39
botocore                    1.15.39
cachetools                  4.0.0
certifi                     2019.11.28
cffi                        1.14.0
chardet                     3.0.4
click                       7.1.1
clyent                      1.2.2
colorama                    0.4.3
conda                       4.8.3
conda-build                 3.18.11
conda-package-handling      1.6.0
conda-verify                3.4.2
cryptography                2.8
cycler                      0.10.0
decorator                   4.4.2
defusedxml                  0.6.0
docutils                    0.15.2
efficientnet                1.1.0
entrypoints                 0.3
fasttext                    0.9.1
filelock                    3.0.12
future                      0.18.2
```

gast	0.2.2
gensim	3.8.1
glob2	0.7
google-auth	1.11.3
google-auth-oauthlib	0.4.1
google-pasta	0.2.0
grpcio	1.27.2
h5py	2.9.0
idna	2.9
imageio	2.8.0
importlib-metadata	1.5.0
ipykernel	5.1.4
ipython	7.13.0
ipython-genutils	0.2.0
jedi	0.16.0
Jinja2	2.11.1
jmespath	0.9.5
joblib	0.14.1
json5	0.9.3
jsonschema	3.2.0
jupyter-client	6.0.0
jupyter-core	4.6.3
jupyterlab	1.2.6
jupyterlab-server	1.0.7
Keras	2.3.1
Keras-Applications	1.0.8
Keras-Preprocessing	1.1.0
kiwisolver	1.1.0
libarchive-c	2.8
Markdown	3.2.1
MarkupSafe	1.1.1
matplotlib	3.2.1
menuinst	1.4.16
mistune	0.8.4

mkl-fft	1.0.15
mkl-random	1.1.0
mkl-service	2.3.0
navigator-updater	0.2.1
nbconvert	5.6.1
nbformat	5.0.4
networkx	2.4
nltk	3.4.5
notebook	6.0.3
numpy	1.18.2
oauthlib	3.1.0
olefile	0.46
opencv-python	4.2.0.32
opt-einsum	3.2.0
pandas	1.0.3
pandocfilters	1.4.2
parso	0.6.2
pickleshare	0.7.5
Pillow	7.0.0
pip	20.0.2
pip-autoremove	0.9.1
pkginfo	1.5.0.1
prometheus-client	0.7.1
prompt-toolkit	3.0.3
protobuf	3.11.4
psutil	5.7.0
pyasn1	0.4.8
pyasn1-modules	0.2.8
pybind11	2.5.0
pycosat	0.6.3
pycparser	2.20
Pygments	2.6.1
PyJWT	1.7.1
pyOpenSSL	19.1.0

pyparsing	2.4.6
pyreadline	2.1
pyrsistent	0.15.7
PySocks	1.7.1
python-dateutil	2.8.1
pytz	2019.3
PyWavelets	1.1.1
pywin32	227
pywinpty	0.5.7
PyYAML	5.3.1
pyzmq	18.1.1
QtPy	1.9.0
requests	2.23.0
requests-oauthlib	1.3.0
rsa	4.0
ruamel-yaml	0.15.87
s3transfer	0.3.3
scikit-image	0.16.2
scikit-learn	0.22.2.post1
scipy	1.4.1
seaborn	0.10.0
Send2Trash	1.5.0
setuptools	46.1.1.post20200323
six	1.14.0
sklearn	0.0
smart-open	1.11.1
soupsieve	2.0
tensorboard	1.15.0
tensorflow-estimator	1.14.0
tensorflow-gpu	1.14.0
tensorflow-gpu-estimator	2.1.0
termcolor	1.1.0
terminado	0.8.3
testpath	0.4.4

tornado	6.0.4
tqdm	4.43.0
traitlets	4.3.3
urllib3	1.25.8
wcwidth	0.1.8
webencodings	0.5.1
Werkzeug	1.0.0
wheel	0.34.2
win-inet-pton	1.1.0
wincertstore	0.2
wrapt	1.12.1
xmltodict	0.12.0
zipp	2.2.0
tesseract-ocr	(4.00~git2288-10f4998a-2).