APPROVAL SHEET

Title of Thesis: Using Text to Improve Classification of Man-Made Objects

Name of Candidate: Akash Alok Vartak
M.S. in Computer Science,
2022

Thesis and Abstract Approved: _____
Dr. Tim Oates
Professor
Department of Computer Science and
Electrical Engineering

Date Approved: _____

Curriculum Vitae

Name: Akash Alok Vartak

Degree and date to be conferred: M.S. in Computer Science, May 2022.

Collegiate institutions attended:

University of Maryland, Baltimore County, M.S. in Computer Science, 2022.
Pune Institute of Computer Technology, BE in Computer Engineering, 2017.

Major: Computer Science

Professional positions held:

Software Analyst; Yardi Software India Private Limited, Pune, India; (July 2017 – Dec. 2018).
Software Developer; Yardi Software India Private Limited, Pune, India; (Jan. 2019 – July 2019).
Project Intern; E2open; (Aug. 2016 – Mar. 2017).

ABSTRACT

Title of Thesis: Using Text to Improve Classification of Man-Made Objects

Akash Alok Vartak,

M.S. in Computer Science, 2022

Thesis directed by: Dr. Tim Oates,
Professor
Department of Computer Science and
Electrical Engineering

People identify man-made objects by their visual appearance and the text on them e.g., does a bottle say water or shampoo? We use text as an important visual cue to help distinguish between similar looking objects. This thesis explores a novel joint model of visual appearance and textual cues for image classification.

We perform this in three functions - (a) Isolating an object in an input image; (b) Extracting text from the image; (c) Training a joint vision/text model. We simplify the task by extracting text separately and presenting it to the model in machine readable format. Such a joint model has utility in many real world challenges where language is interpreted through a sensory perception like vision or sound.

The aim of the research is to understand whether visual percepts, when understood in the context of extracted language, will provide a better classification of image objects than using only pure vision to perform image classification. In conclusion, we show that joint classifier models can successfully make use of text present in images to classify objects, provided that the extracted text from images is of high quality and we have the number of images proportional to the number of classification classes.

Keywords: Computer Vision, Classification, Deep Learning, NLP, Text Extraction

Using Text to Improve Classification of Man-Made Objects

by

Akash Alok Vartak

Thesis submitted to the Faculty of the Graduate School
of the University of Maryland in partial fulfillment
of the requirements for the degree of
M.S. in Computer Science
2022

Dedicated to my family, especially my Mom, who encouraged me to put in my 100% everyday.

ACKNOWLEDGMENTS

I would like to thank my advisor and mentor, Dr. Tim Oates, for his amazing guidance and for showing immense confidence in me throughout my time at UMBC and in my research. It has been a privilege to work with him and to learn from him. I would like to thank Ashwinkumar Ganesan for his constant help. His insights into experiment results were key for me to make progress in my research. I would also like to thank Dr. David Chapman and Dr. Tim Finin for agreeing to be on my committee.

I would like to express my deepest gratitude to my parents, Mrudula and Alok Vartak and my sister Maithili Vartak for being a pillar in my life and for their unfailing encouragement.

A special thank you to Akshay Peshave for being an amazing friend, making my research experience fun and for being a constant source of new information. Last but not the least, I want to thank all my friends here in CORAL Lab, in USA and back at home in India who made my Masters journey so memorable and enjoyable. You guys are my family away from home.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Research in Artificial Intelligence (AI) has been accelerating for the past decade and a half, with significant work in multi-modal systems. There has been a renewed interest in one such discipline - image captioning and textual reasoning in the context of images. This is an important AI sub-system that encompasses Computer Vision (CV) and Natural Language Processing (NLP). This is because CV and NLP are crucial to the advancement of AI as they model two of human beings' most important faculties - vision and language.

Human beings see an object and instantly know what it is, due to their powers of superior perception and innate ability to understand objects in the context of their surroundings, which is not the case for machines. Machines often need to understand images without any context. We hypothesize that text found on objects in the real world will offer great insight into the object itself and improve the accuracy of a classification model. Through this research we propose a model that can take an image, extract text from that image, reason about the image in context of the text, and classify that object.

There are a number applications of such an integrated pipleline. It can be used in online retail marketplaces to immediately classify objects by type and category, for users and buyers to quickly sort through. It can be used to model visual reasoning systems in autonomous cars to understand street signs for navigation.

We have proposed a novel architecture that combines a text extraction tool and a joint-classifier to create an end-to-end pipleline for image classification in context of the language in the image. The joint-classifier is an amalgamation of an image classification model (a convolutional neural network), and a sequence learning model (a Long Short Term Memory). We briefly talk about the past research in this area, and then describe in detail the dataset we compiled and the experiments we formulated and ran. Next, we explain the experiment results, and interpret the results and their impact on our hypothesis. Finally, we present our conclusion and provide ideas for future work.

Chapter 2

BACKGROUND AND RELATED WORK

In this chapter we describe the required background and prior work related to our research.

2.1 Background

2.1.1 Residual Networks

Ever since the advent of convolutional neural networks (CNN), the best results on non-trivial vision-based tasks are accomplished using 'deep' or 'very deep' models. This established the significance of large depth as one of the most important factors in designing and choosing a CNN of a particular architecture. But with increasing depths came the problem of 'vanishing gradients'. This occurs when the weights of each previous layer of a network receive corrective updates that are proportional to the partial derivative of the error function of weights in the current layer. This causes each subsequent update's magnitude to decrease. In some cases it is so minute that it prevents the model from learning or converging (Hochreiter 1998).

The vanishing gradient problem was then addressed by using normalized initialization and intermediate normalization which allowed deep networks to converge with back-propagation. This brought the problem of 'degradation', where by increasing depth net-

works were able to get better accuracy, but this accuracy saturated and after a point started decreasing or degrading.

In the paper "Deep Residual Learning for Image Recognition", He et al. proposed a convolutional neural network model known as the Residual Networks or 'ResNet' (He *et al.* 2015). It is made of multiple building blocks that are characterized by the equation $F(x) + x$ as shown in Figure 2.1, where the networks are "feed-forward networks with shortcut connections" (He *et al.* 2015). These shortcuts skip one or more layers and are, in essence, identity mappings whose outputs are added to the outputs of the skipped layer and fed to the next layer.

Formally, the building blocks are defined as

$$y = F(x, \{W_i\}) + x \tag{2.1}$$

where x and y are input and output vectors of the layers and $F(x, \{W_i\})$ is the residual mapping which has to be learnt. The additive operation is a straight forward element-wise addition. The shortcut connections introduce no new parameters and are computationally non-complex. This allowed the authors to compare the residual networks and their plain network counterparts. It is crucial that dimensions of x and F are equal. If not, then a linear projection shortcut W_s is required:

$$y = F(x, \{W_i\}) + W_s x \tag{2.2}$$

(He *et al.* 2015)

The authors were able to show that their networks, with skip connections, had lower training error than similar networks without skip connections, and can continue to be trained by the same back-propagation algorithm to get consistently increasing accuracy

even at higher depths.

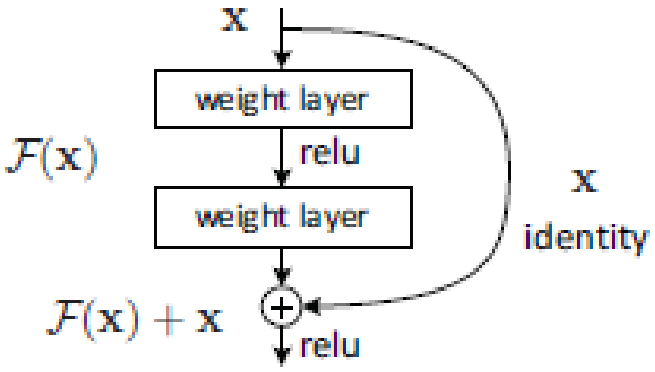


FIG. 2.1: Residual learning block (He *et al.* 2015).

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

FIG. 2.2: Original table showing the ResNet architectures (He *et al.* 2015).

2.1.2 ResNet-18

ResNet-18 is a residual network of 18 layers and is inspired from the VGG network models. It takes as input a 3-channel (RGB) image of size 224x224 and ends with a 1000-way fully connected layer output.

Each convolution layer uses a 3x3 filter, stride of 1, and a fixed feature map dimension of 64, 128, 256, and 512. Shortcut connections are shown by solid curved lines in Figure 2.3 and are used to bypass the input every 2 convolutions. In a single layer, the height and width dimensions remain constant. These shortcuts, as defined by Equation 2.1, are used when the input and output are of the same dimension size. When the required dimension increases, ResNet either performs a padding with extra zeros to increase the input dimension or uses a projection shortcut as shown in Equation 2.2.

When there is a change in the feature map size (size is halved), the number of filters is doubled. He et al. state that this is to preserve the time complexity of each layer. To achieve this result, the ResNet architecture performs a convolution with a stride of 2 as shown by the dotted curved lines in Figure 2.3. This is the fist convolution of each layer instead of the more commonly used max-pooling with a stride of 2.

All original ResNet architectures are shown in Figure 2.2

2.1.3 Long Short Term Memory (LSTM)

Long Short Term Memory (Hochreiter & Schmidhuber 1997) networks are special neural networks that are designed to learn sequences and are capable of learning long-term dependencies. LSTM's core behaviour is structured around remembering and retaining information for long periods of time. All such sequence learning networks have a chain-like architecture of repeating modules of the network. In LSTMs, the repeating module is a single block of 4 interacting layers as shown in Figure 2.4. The blue items are layers and the yellow items are point-wise operations.

The key of an LSTM's operation is the cell states - the top horizontal line in Figure 2.4, going from C_{t-1} to C_t. The cell state is what holds the information flowing through the LSTM network, to which information can be added or removed via 'gates'. Gates are operations that optionally let information through or 'forget' unwanted information. A

single LSTM cell functions as follows (all symbols and notations correspond to the ones in Figure 2.4):

- First, the LSTM decides what information it does not need to retain. This decision is made by the 'forget gate'. It is characterized by the equation:
  $f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f)$
  h_{t-1} is the hidden state of the previous cell and is the input to the current cell. x_t is the information that the current cell needs to process.

- The next step is to identify what information needs to be stored in the cell state and this is done in 2 parts. First the 'input gate' decides what to update and then the 'tanh layer' creates a new vector of candidate values C'_t. These will be updated to then be added to the cell state. This step is characterized by the equations:
  $i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$
  $C'_t = tanh(W_C.[h_{t-1}, x_t] + b_C)$

- Next, the cell state needs to be updated from C_{t-1} to C_t. This is done via point-wise multiplication and addition operations as follows:
  $C_t = f_t \times C_{t-1} + i_t \times C'_t$

- Now, the new cell state is available and the LSTM must decide what information to output. For this, a $tanh$ function is used to push values of the output within -1 and 1 and is then multiplied to the output of the sigmoid gate. This gives us the next cell's hidden state:
  $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
  $h_t = o_t \times tanh(C_t)$

2.2 Related Work

Research in the field of text based reasoning, in context of objects and images, has been developing for quite some time. Over the years, many tasks have been proposed to model various human abilities such as identifying objects in images (Krizhevsky, Sutskever, & Hinton 2017), describing an object or scene in an image (Lin *et al.* 2014), learning to recognize novel objects that are not present in the dataset used to train the model (Hendricks *et al.* 2015), the ability to answer questions about an image (Antol *et al.* 2015) and even generating imaginary stories about sequence of images (Huang *et al.* 2016).

One of the main sub-fields that papers and researchers tackle in this domain is that of VQA - Visual Question Answering. VQA involves a set of images of objects with text and then answering a question about that text in context of the object or vice versa. A novel approach to this task was proposed by (Singh *et al.* 2019). They proposed a model LoRRA - Look, Read, Reason and Answer - that reads text, reasons about it in the context of the image and predicts an answer for the question asked. They built their own 'TextVQA' dataset comprising of 45336 questions on 28408 images. These questions generally require reasoning about the text in order to satisfactorily answer questions, and explained that any VQA model actually needs to:

- Understand what question is being asked about the image and text.
- Find image regions that contain text and convert those regions into text that the predictor model can understand.
- Try and jointly reason about this text and image with respect to the question.
- Predict whether the image's text contains an answer to the question or if the answer needs to be inferred from the joint reasoning.

This shows the complexity and sheer volume of the sub-tasks that a VQA model needs to complete. For this, they were the first to propose the use of Optical Character Recognition (OCR) as a module in a VQA model. However, their model was limited in scope to OCR.

Another field of research is that of Scene Text Recognition (STR). STR refers to the task of reading text from natural scenes or images of natural world objects. The stimulus received by text recognition tasks was influenced by the maturity of OCR models. OCR could easily be applied to cleaned documents and manuscripts to extract the text that the specific OCR model was programmed and trained on. But even mature OCR methods failed on STR tasks due to the diverse nature of texts and their varied appearances in the real world. Baek et al. 2019 studied many such STR models and their corresponding datasets. They examined inconsistencies in the training and evaluation datasets and resultant performance drops. The datasets they compared ranged from synthetically generated word sets like MJSynth and SynthText to real-world datasets like Street View Text (SVT) and IIIT5K-Words. They then proposed a multistage STR model that found image regions containing text, normalized these regions, and performed feature extraction and text prediction. Their model was a deep learning based text recognition technique that did well on a myriad range of real world texts.

We go a step beyond all prior work and propose a joint classifier that combines an image classifier and a text classifier. It jointly learns to associate the text found in images with the object in the image itself, and then classify the object into a category.

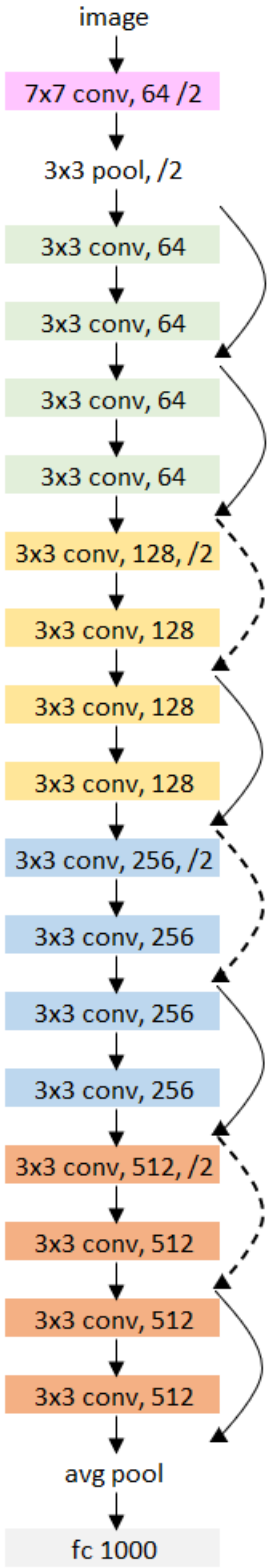


FIG. 2.3: ResNet-18 model architecture visualized.

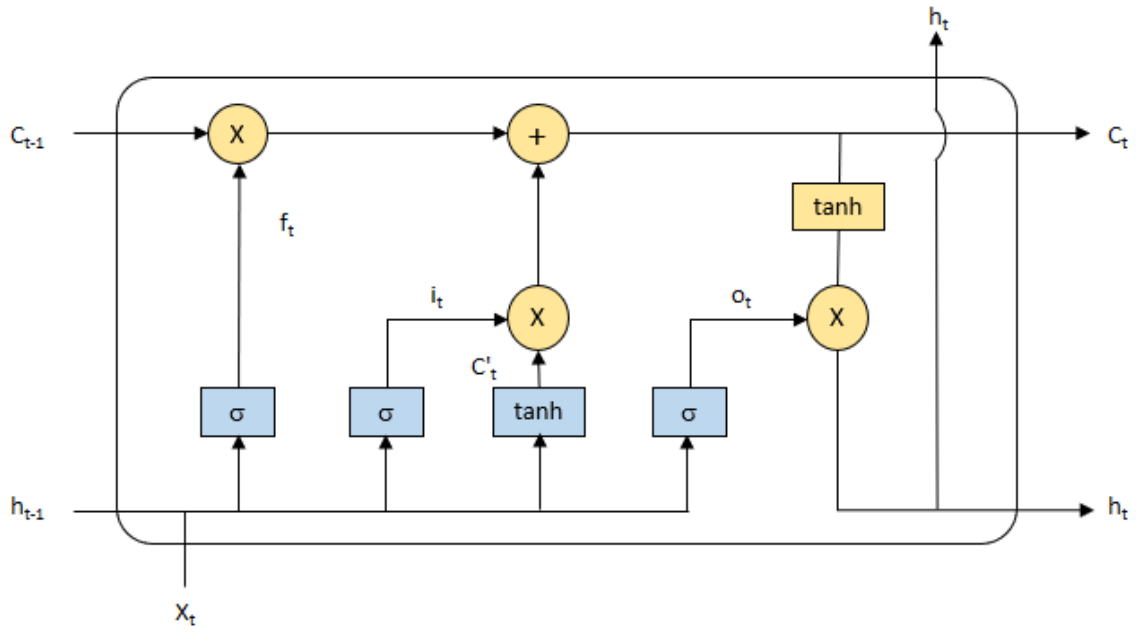


FIG. 2.4: A single repeating block of an LSTM.

Chapter 3

DATASET

This chapter explains how we created our own Amazon Product dataset, using the publicly available metadata files from the Amazon Review Dataset (Ni, Li, & McAuley 2019).

3.1 Amazon Review Dataset

The Amazon Review Dataset, published by Ni, Li, & McAuley, is made free-to-use for research purposes. We use the metadata files as our initial source CSVs. These CSVs provide various features like product category, product description, similar 'also bought' and 'also viewed' products, brand name, and product identification 'ASIN' number.

We use only 6 categories as they provide us with a sufficiently large dataset to work on. The categories used are - Appliances, Luxury Beauty, Magazine Subscriptions, Software, Musical Instruments, and Gift Cards. Sample images from the dataset are shown in Figure 3.1 and Figure 3.2. The original Amazon Review Dataset does not contain any images and has only the aforementioned product metadata. The images were scraped as explained in the following sections.

3.2 Our Amazon Product Dataset

3.2.1 Product Images Scraping

Each category's metadata CSV has multiple products, and each product line provides the product ASIN which can be used to uniquely identify it. We wrote a Python Selenium-based image scraper that uses the ASIN, and generates and opens an Amazon marketplace URL. The scraper then clicks each thumbnail image of the product and downloads the images that get displayed on an Amazon product page.

Most Amazon products have multiple images per product, but some products have no images uploaded on the products page. We ignore such products as they cannot be used in our experiments. As each image is being scraped, a new JSON file is created that contains the original metadata filename, product's ASIN, category and file path of the saved image. This automated image scraping process is repeated for all product listings for the 6 aforementioned categories. Each category has its own independent individual JSON of saved images with a filename format of 'scraped_meta_⟨category⟩.json'. Examination of the scraped images revealed that there is an imbalance in the dataset - each class has a different number of images.

3.2.2 Text extraction via Tesseract OCR

For text extraction, we used Tesseract as the OCR. The JSONs created during image scraping are the input to this phase.

Tesseract, for text extraction, can be configured on a wide variety of parameters like image resizing percentage, Mean Blurring aperture size, Adaptive vs Binary thresholding and Mean vs Gaussian thresholding methods, among others. After experimenting with a host of various combinations, we settled on the following combination: images would be resized to 190%, median blurring would be employed with an aperture size of 3 (must be

an odd number) and binary thresholding (Otsu method).

In an automated python script, for each image a thread is spawned that does the following tasks in-order: open image and convert to grayscale, resize image to 190%, perform binary thresholding on resized image, apply median blurring, and save this as a temporary image. This saved image is then passed as input to Tesseract and it extracts text which is saved in a CSV that contains the original image path, the extracted text, and the product category. The extracted text is converted to lowercase for the sake of uniformity. Similar to the JSONs, each category has its individual CSVs of extracted texts with a filename format of 'text_extracted_meta_⟨category⟩.csv'.

3.2.3 Final Dataset Consolidation

Our final dataset consists of 2 components - a single CSV containing all the extracted text, and the scraped images. The single final dataset CSV is created by simply concatenating all individual CSVs into one. Both these dataset components will be used as inputs to both the two baseline and combined classifiers in the experiments, as explained in the next chapter.

FIG. 3.1: Sample dataset images for each category along with possible text for OCR to extract. Categories starting horizontally from top left: Appliances, Luxury Beauty, Magazine Subscriptions, Software, Musical Instruments, and Gift Cards. (Ni, Li, & McAuley 2019).

FIG. 3.2: Sample dataset images for each category that contain no readable text. Categories starting horizontally from top left: Appliances, Luxury Beauty, Musical Instruments, Gift Cards and Software. 'Magazine Subscriptions' category does not have any images without text (Ni, Li, & McAuley 2019).

Chapter 4

ARCHITECTURE, EXPERIMENTS AND RESULTS

4.1 Architecture

The final goal of this project is to create a complete integrated pipeline consisting of: (1) automated text extraction by an OCR tool and (2) a classification model. The classification model would be an ensemble of two parts in sequence - a ResNet18 and an LSTM. The model would take object images that contain text as input and classify these objects in 1 of the 6 categories - Appliances, Luxury Beauty, Magazine Subscriptions, Software, Musical Instruments, and Gift Cards. This pipeline is shown in Figure 4.1. Yellow items denote components of the pipeline, blue items are inputs to the components and the green item denotes the predicted category of the image. We hypothesize that the textual cues in the images would aid in classification of the object (in the image), and that the extracted text would be especially useful if it were reasonably good or near perfect, e.g., as a human reader would extract it.

Before we can build a combined classifier we create two baseline classifiers. Baseline-1 is an image-only classifier that performs classification purely on image inputs and is a ResNet-18 convolutional neural network. Baseline-2 is a text-only classifier that performs classification purely on the OCR extracted text. These baseline classifiers are intended to give us an idea of how the individual models fare on classifying the object images and how

they contribute in the combined classification model.

First, the dataset was randomized to prevent any two runs of the experiment from using the same train, validation and test example sequences. It was then partitioned to the respective sets. This randomization was done to help minimize the chances of having a biased dataset and increase the probability of having a representative dataset for training. For each experiment, the dataset was divided in ratios of 0.7:0.15:0.15 to separate into train, validation, and test datasets. Due to the unique nature of our model that required it to have two inputs - image and text - we implemented our custom collate function to create input batches for the PyTorch based model.

In each run of any classification model, if the model used image inputs, the images were resized to ResNet's input size of 224x224 and center cropped to get a centered, squared image. They were then normalized around the mean and standard deviation of ImageNet dataset values - mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]. ImageNet values were used because the images in our dataset are photos of objects in "natural scenes".

For all textual data, we used word embeddings as input to the LSTM. The FastText's Wiki News 'wiki-news-300d-1M' word embeddings corpus was used (Mikolov *et al.* 2018).

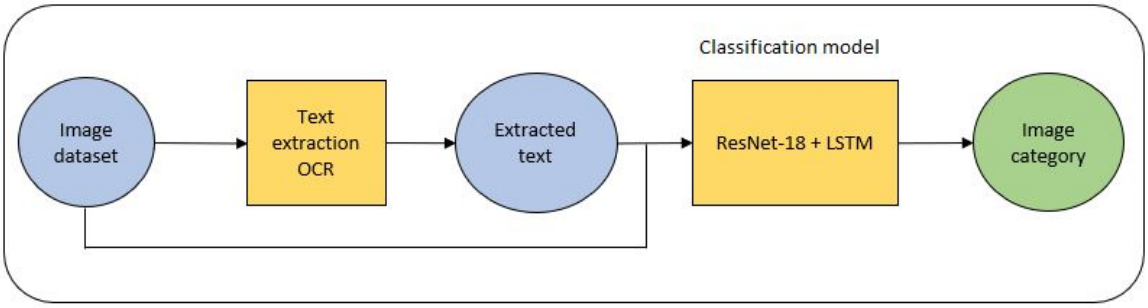


FIG. 4.1: The final architecture pipeline.

4.2 Experiments

For every experiment, we ran the specific experiment's dataset through each of the two baseline classifiers as well as the combined classifier. Below, we have explained the dataset or sub-dataset used for each experiment, experiment parameters, classification results and our interpretation as to how they affect our hypothesis. Note: in every experiment, the model considers only those image input examples that have some extracted text and ignores those images that have no extracted text. Only about 25% of the images had some text extracted from them.

Our image based classification model uses ResNet-18 as the base pretrained network, and replaces its fully-connected layers by our hidden fully-connected layer. Our text based classification model is an LSTM with an input word embedding with a dimension of 300 (number of LSTM's expected features). It is a stacked LSTM that employs 2 recurrent layers.

We used the following standard parameters through each experiment:

- training batch size of 64
- validation and test batch size of 1000
- learning rate of 0.001
- 25 epochs for train, validation and test
- word embedding dimension of 300

4.2.1 Unbalanced Classification

We used our complete dataset for this experiment. Since each classification model considers only images with some extracted text, the dataset for this experiment was of

24218 images, which was then partitioned into 16952, 3632, 3634 images respectively in train, validation, and test sets.

The experiment's goal was just to see how each classifier would fare on our dataset, and understand whether we needed to expand the dataset or rerun text extraction using a better OCR, and further to see how the dataset's unbalanced class distribution would affect the classifiers.

Results for this experiment are shown in Table 4.2a and graphically in Graph 4.2b. At first glance, it seems as if the text-only classifier is independently performing nearly as good as the independent image-only classifier. But, on closer examination of the text-only classifier's confusion matrix (shown in Table 4.3a), we see that it achieves a 70%+ accuracy since it predicts most instances to be the frequently occurring class - 'Luxury Beauty'. This prompted an examination of the image-only classifier's confusion matrix (shown in Table 4.3b), and it too revealed the same - it predicts most instances to be the frequently occurring class.

Due to this nature of each baseline model, the combined classifier also takes a similar approach and this behaviour does not provide any insight about the usefulness of the extracted text.

Our dataset is inherently unbalanced, and since we consider only images with some extracted text, this imbalance is further accentuated. These results, coupled with the dataset's imbalance, make it clear that the classifiers are behaving as a dominant class predictor. This neither helps to prove nor disprove our original hypothesis, since it does not provide any evidence as to how textual data is affecting the classification.

4.2.2 Unbalanced Classification with Normalized Weight Balancing

As observed in the earlier experiment, classification on unbalanced data leads the classifier to predict the dominant class. To fight this imbalance, we decided to use the

same dataset, but this time we introduced weight normalization. We counted the number of images per class and gave each class a weight that is inversely proportional to the number of images per class - higher weight to the least frequent class and least weight to the most frequent class. This was done as shown by Algorithm 1.

Algorithm 1 An algorithm to calculate class weights.

Require: 'label_sample_count' is a dictionary that contains count of number of images per class.
Ensure: $label\_weights$ will contain final normalized class weights.

$counts \leftarrow list[label\_sample\_count.values()]$
$label\_weights\_temp \leftarrow []$ ▷ Calculate class weights.
for c in $counts$ **do**
  if $c \neq 0$ **then**
    $label\_weights\_temp.append() \leftarrow \frac{sum(counts)}{c}$
  else
    $label\_weights\_temp.append() \leftarrow 0$
  end if
end for
$label\_weights \leftarrow []$ ▷ Normalize class weights.
for lx in $label\_weights\_temp$ **do**
  $label\_weights.append() \leftarrow \frac{(lw - min(label\_weights\_temp)}{(max(label\_weights\_temp) - min(label\_weights\_temp))}$
end for

Following this, the dataset was split in the same way as before - partitioned into 16952, 3632, 3634 images each in train, validation, and test sets respectively.

This experiment was specifically designed to see if classification improved when these weights were assigned to classes. The class weights come into play when we calculate a model's prediction loss in training mode. Results for this experiment are shown in Table 4.4a and graphically in Graph 4.4b.

Results show that the image-only classifier continues to predicts classes with a satisfactory accuracy. Examination of the image-only model's confusion matrix (Figure 4.6) shows that it benefited from the addition of class weights and was no longer behaving as a

dominant class predictor.

However, our text-only classifier accuracy drops significantly even after using class weights. This could be attributed to the poor quality of input data, which in our case is the extracted text. This was proven by an examination of the extracted text and led to the following finding. Either:

- Text was not extracted in its entirety - some words from the text were not extracted.
- In other cases, the extracted text had words no longer than 2-3 letters each and/or were not valid English words.

An example image of this issue is shown in Figure 4.5. The OCR's extracted text for this image was "McGraw-Hill LearnSmart® gets you BAe the grade you've atways wanted. Se leerinan tne oo nen a an SAS rete Ne RA POUT OR Cire One Bert DerES poten a HV gu Wel were mate (er tunpet". This clearly shows that only a small percentage of the text from the complete extraction could be useful to a human classifier. This example has all of the above issues - some text is not extracted and the part that is extracted is either incorrect or not understandable. Even in the case of clearly readable text in the image, the extraction was poor - the first sentence of the extraction should have been close to: 'McGraw-Hill LearnSmart gets you the grade you've always wanted'. Consequently, it can be said that the text-only classifier was unable to get any information from the available language.

Our combined classifier's accuracy also suffered due to this inability to generalize on the poor quality textual data. As seen in the Table 4.4a, the combined classifier's accuracy averages at just 63%. Thus, our hypothesis, again, could not be proven or disproven. We tried to tweak Tesseract OCR to improve the quality of the extraction, but this did not give satisfactory extraction results. We then moved on to the next experiment where we try to mitigate these text issues.

4.2.3 Balanced Classification

This section describes the experiments conducted on balanced datasets wherein each class had the same number of images. Similar to earlier experiments, every experiment's dataset was first randomized and then split in train, validation and test sets.

24-image Subset Experiment

In the earlier experiment, we applied normalized class weights to diminish the effect of the dataset's imbalance and to improve the accuracy of our classification models. Instead, we saw that though the image classifier benefited from these class weights, the text-only and subsequently the combined classifier did not. This was due to low quality text being extracted by the OCR.

The 24-image subset experiment was designed to solely test whether even small amounts of perfect/near perfect text allowed the text-only classifier to aid in object classification. To first improve the text quality we decided to manually hand-extract text from 24 images from each category. This would constitute our dataset and this set number of images from each category also removes any dataset imbalance. The resultant dataset was made of 192 images and was then split into 134, 28 and 30 images respectively in train, validation and test sets.

The models' accuracy results for this experiment are shown as in Table 4.7a and graphically in Graph 4.7b.

The graph shows that the accuracies of all 3 models dropped to below 20%. This is due to an extremely small dataset size of just 192 images in total. This accuracy drop can be understood intuitively - since the models did not have enough data to train on it led to poor generalization. Additionally, the accuracy drop cannot be attributed to over-fitting due to the small dataset size. The evidence to prove this is shown in Figures 4.8, 4.9, 4.10. They show that neither train, validation nor test witnessed any accuracy drops till the end of the

experiment.

Having said that, it can be seen that improved text quality helps the text-only classifier fare better than before. It can be said that even small amounts of good text help the text-only classifier improve accuracy. Consequently, the combined classifier also benefited from good extracted text and had an accuracy that was not too far from that of the two baseline classifiers.

Due to human limitations we could extract text for only 24 images of each category but we believe that if this is done on a larger scale (at least 1000 images per category) either by hand or by an improved OCR tool, we shall see the hypothesis holding - 'good' textual language will aid in classification of objects.

75-image Subset Experiment The previous experiment showed that textual cues do certainly have the potential to aid in classification of objects, but in that experiment we had a large number of classes and not enough training samples in proportion. We also saw that even a small amount of good textual data is useful for the classifier. We wanted to build on both these notions and reduce the number of classes and increase the number of samples per class. We devised the 75-image subset experiment to test whether, given a subset of classes and a proportionate number of training samples with the same kind of near-perfect extracted text, the classifier can achieve better accuracy. This would help prove that text can improve classification accuracy.

For this, we decided to use two classes - Luxury Beauty and Magazine Subscriptions - and hand-extracted text from 75 random images of each category. Our dataset was of 150 images in total which was then split into 105, 22, and 23 images respectively in train, validation, and test sets.

The accuracy results of this experiment are shown in Table 4.11a and Graph 4.11b.

We can see that model accuracies have risen considerably due to number of images

being proportional to the number of classes. Additionally, the text-only classifier has an accuracy that is equivalent to that of the image-only classifier. However, the accuracy was still in the range of 50-70%. This warranted further examination of the images and text.

We saw that images within a category are not one type of image. For example, in 'Luxury Beauty', images are not standardized as always a bottle or a tube, but rather they are a variety of images and with a dataset of size 150, the image-only classifier has a tough time trying to learn from the limited training set. The problem for the text-only classifier is that the text is not unique enough to classify between categories. Both categories - 'Luxury Beauty' and 'Magazine Subscriptions' had common words like 'health', 'beauty' and 'easy'. Having unique words for clean or even reasonably good classification is necessary since word-level embeddings were used.

Thus, it is clear that perfect/nearly perfect textual data helps the text-only classifier and in-turn helps the combined classifier. But, this textual data needs to be unique enough to have category-specific feature markers that point to which category that text belongs to.

These issues affected the combined classifier too but the quality of the extracted text coupled with a pretrained ResNet, we achieved an accuracy higher than both the baseline classifiers.

This helps nudge us towards thinking that our hypothesis is true - that 'given a subset of classes and enough training samples in proportion with the same kind of near-perfect extracted text' the baseline classifiers and combined classifier can give good classification results. This shows that when high quality text extracted from images is combined with the image itself, it does have the potential to improve classification accuracy.

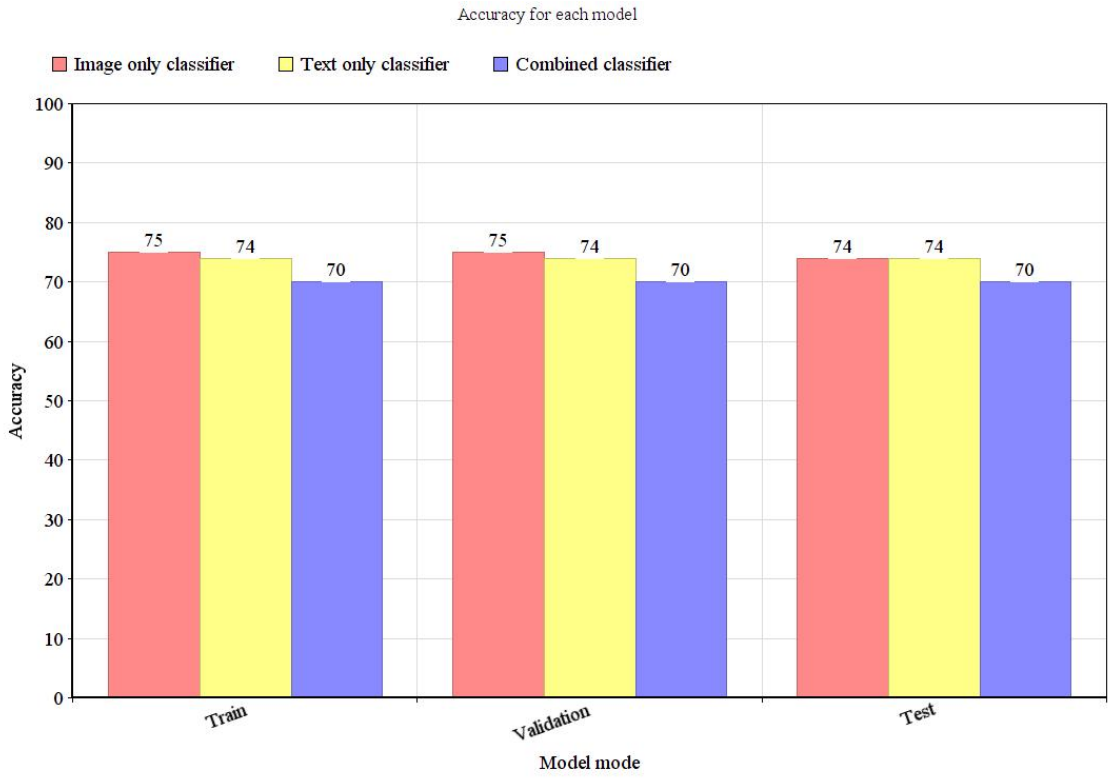
The rise of accuracies of each model, relative to the previous experiment, can be explained by another factor. This experiment performs classification by using 2 categories instead of all 6 and focuses on increasing the amount of perfect text available per image per category. From the perspective of a binary classification problem, the dataset size is a

decently large 75 images per category compared to the previous experiment's 24 images. Consequently, the joint classifier had an easier time learning and generalizing on binary classes rather then multi-class.

Thus, the conclusion can be re-framed to say that when high quality text extracted from images is combined with the image itself, and when the quantity of high quality extracted text is proportional to images per category, it does have the potential to improve classification accuracy.

Classifier Name	Dataset Size	Accuracy (%)		
		Train	Validation	Test
Image Only	24218	75	75	74
Text Only		74	74	74
Combined		70	70	70

(a) Table of classification accuracies.



(b) Plot of classification accuracies.

FIG. 4.2: Classification accuracy of each model for unbalanced classification experiment.

Luxury Beauty	2695	0	0	0	0	0
Magazine Subscriptions	321	0	0	0	0	0
Software	41	0	0	0	0	0
Musical Instruments	481	0	0	0	0	0
Gift Cards	17	0	0	0	0	0
Appliances	79	0	0	0	0	0
	Luxury Beauty	Magazine Subscriptions	Software	Musical Instruments	Gift Cards	Appliances

(a) Confusion matrix for the Text-only classifier.

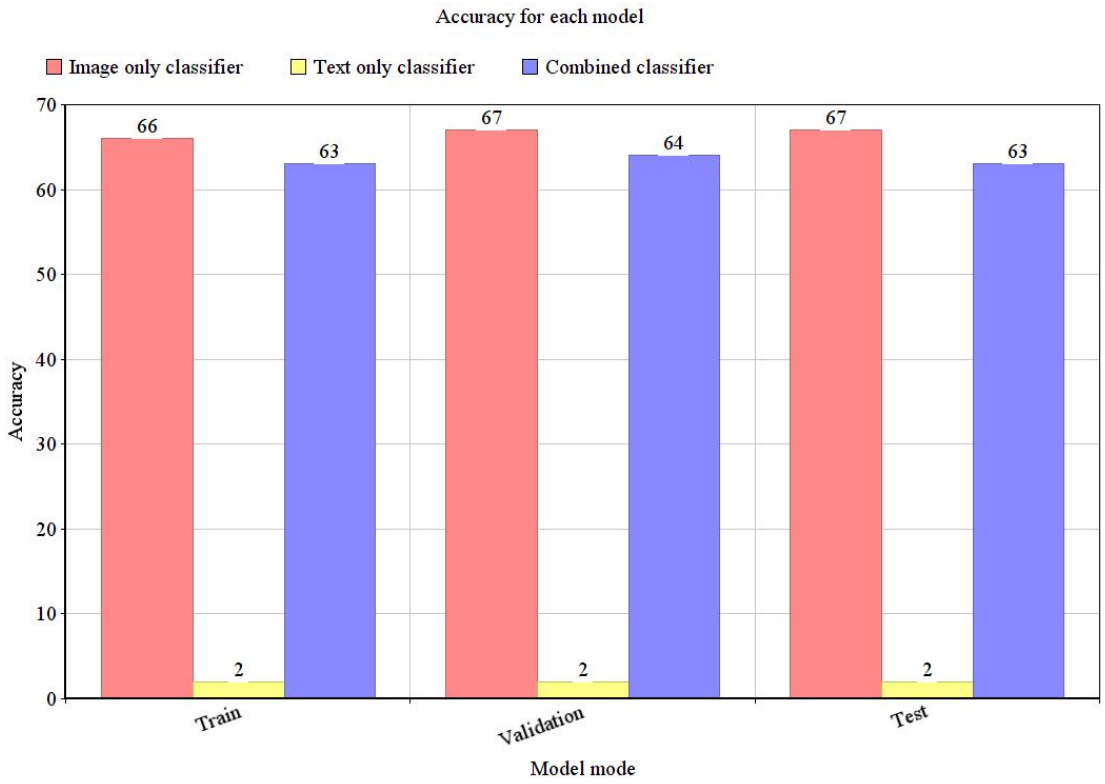
Luxury Beauty	2694	0	0	1	0	0
Magazine Subscriptions	298	19	0	4	0	0
Software	41	0	0	0	0	0
Musical Instruments	469	1	0	11	0	0
Gift Cards	16	1	0	0	0	0
Appliances	78	0	0	1	0	0
	Luxury Beauty	Magazine Subscriptions	Software	Musical Instruments	Gift Cards	Appliances

(b) Confusion matrix for the Image-only classifier.

FIG. 4.3: Confusion matrices for the baseline classifier models for unbalanced classification experiment. True labels are on the Y-axis are the predicted labels are on the X-axis.

Classifier Name	Dataset Size	Accuracy (%)		
		Train	Validation	Test
Image Only	24218	66	67	67
Text Only		2	2	1
Combined		63	64	63

(a) Table of classification accuracies.



(b) Plot of classification accuracies.

FIG. 4.4: Classification accuracy of each model for unbalanced classification with normalized weight balancing experiment.

FIG. 4.5: Example image for text not extracted properly.

Luxury Beauty	0	0	2695	0	0	0
Magazine Subscriptions	0	0	321	0	0	0
Software	0	0	41	0	0	0
Musical Instruments	0	0	480	1	0	0
Gift Cards	0	0	17	0	0	0
Appliances	0	0	79	0	0	0
	Luxury Beauty	Magazine Subscriptions	Software	Musical Instruments	Gift Cards	Appliances

(a) Confusion matrix for the Text-only classifier.

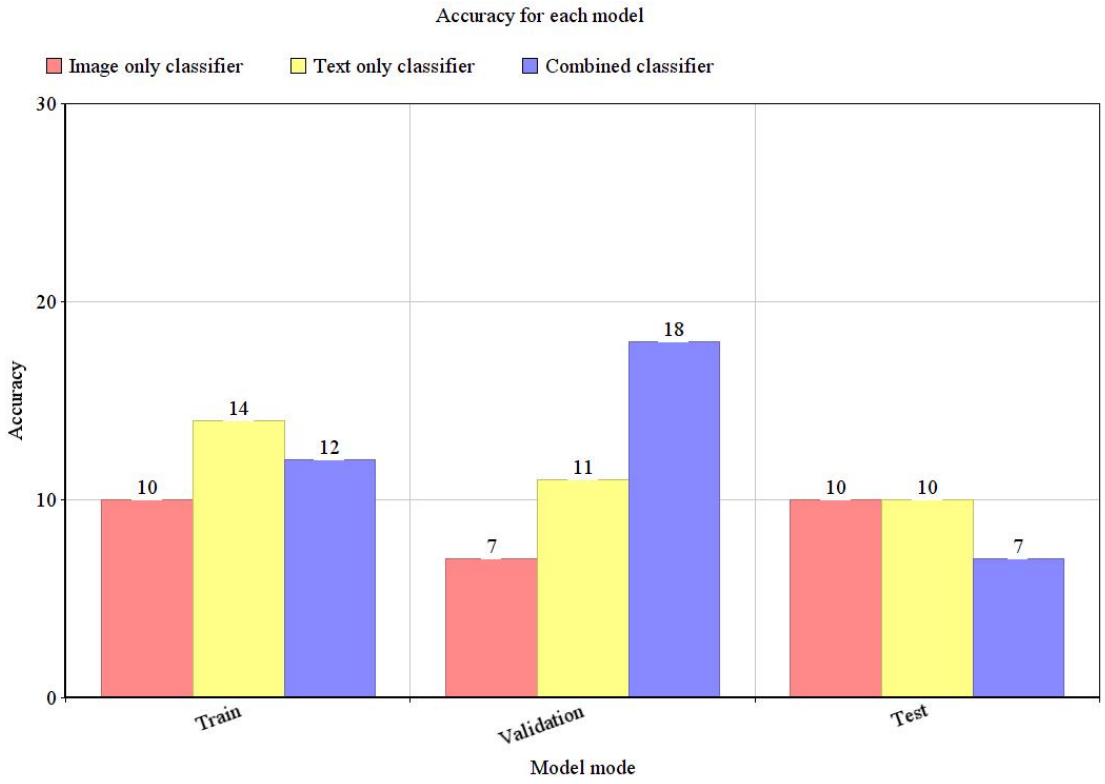
Luxury Beauty	1770	183	29	614	0	99
Magazine Subscriptions	2	308	5	4	0	2
Software	2	18	9	9	0	3
Musical Instruments	38	71	22	305	0	45
Gift Cards	4	7	1	4	0	1
Appliances	6	14	5	27	0	27
	Luxury Beauty	Magazine Subscriptions	Software	Musical Instruments	Gift Cards	Appliances

(b) Confusion matrix for the Image-only classifier.

FIG. 4.6: Confusion matrices for the baseline classifier models for unbalanced classification with normalized weight balancing experiment. True labels are on the Y-axis are the predicted labels are on the X-axis.

Classifier Name	Dataset Size	Accuracy (%)		
		Train	Validation	Test
Image Only	192	10	7	10
Text Only		14	11	10
Combined		12	18	7

(a) Table of classification accuracies.



(b) Plot of classification accuracies.

FIG. 4.7: Classification accuracy of each model for the 24-subset experiment.

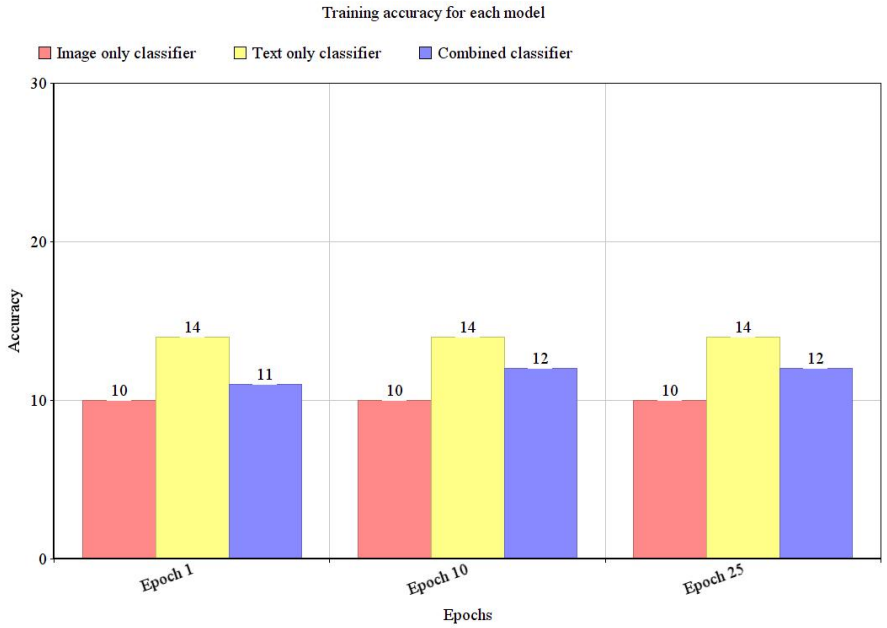


FIG. 4.8: Train accuracy snapshots at various epochs for the 24-subset experiment.

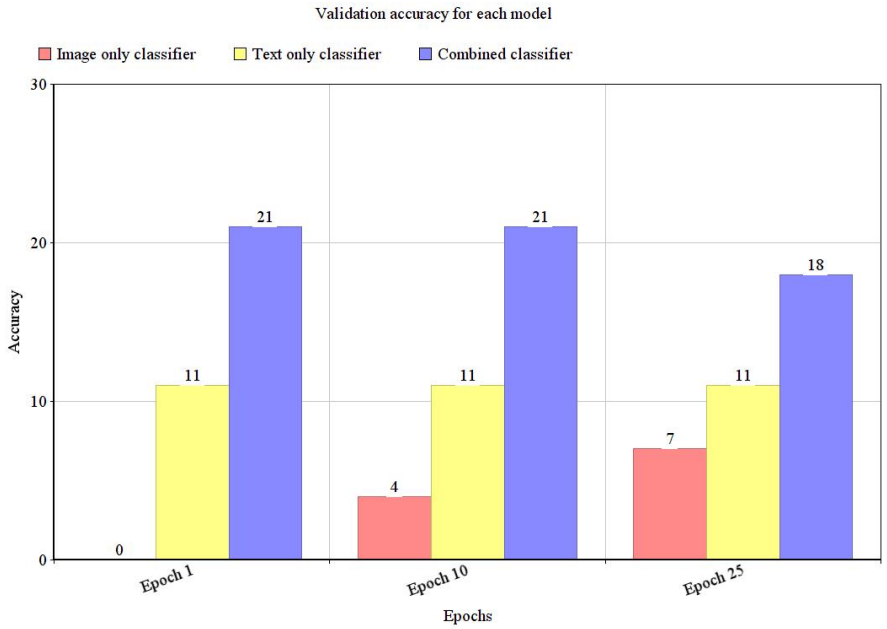


FIG. 4.9: Validation accuracy snapshots at various epochs for the 24-subset experiment.

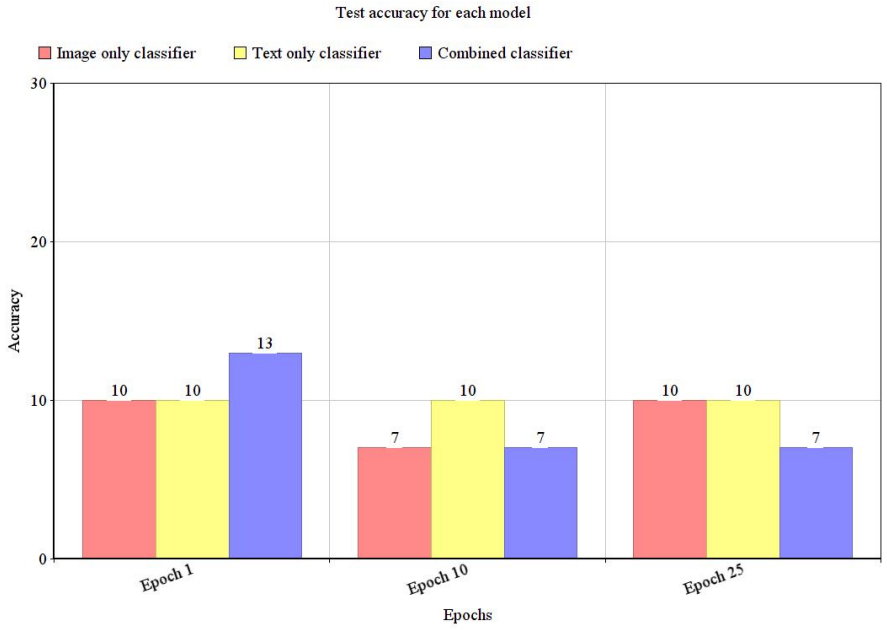
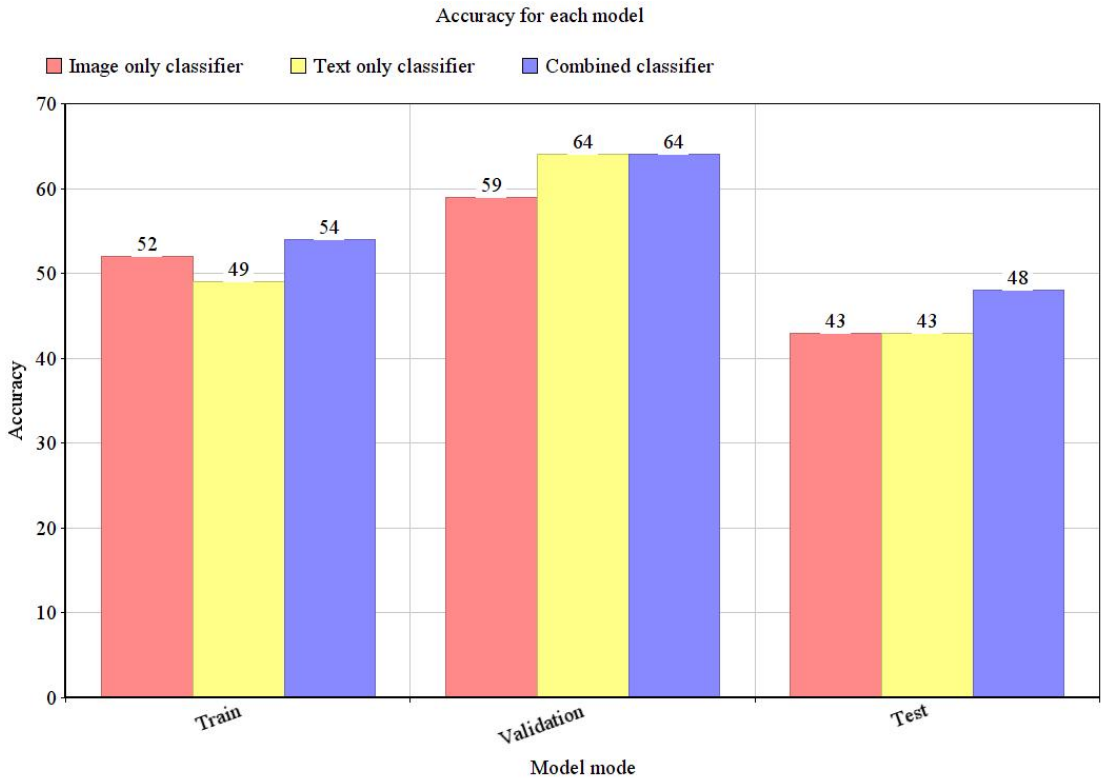


FIG. 4.10: Test accuracy snapshots at various epochs for the 24-subset experiment.

Classifier Name	Dataset Size	Accuracy (%)		
		Train	Validation	Test
Image Only	150	52	59	43
Text Only		49	64	43
Combined		54	64	48

(a) Table of classification accuracies.



(b) Plot of classification accuracies.

FIG. 4.11: Classification accuracy of each model for the 75-subset experiment.

Chapter 5

CONCLUSION AND FUTURE SCOPE

In this research we initially stated our hypothesis that when classifying images of objects, the text found on objects coupled with the image of the object itself would give a more accurate classification of the object. These images we used were images of products on Amazon (Ni, Li, & McAuley 2019) and we extracted text from images using an automated tool as well as manually by hand.

We then went on to design, execute and demonstrate experiments that were aimed at proving this hypothesis. We used ResNet and LSTM to build a combined classifier that would classify images using images as well as the extracted text. To understand how the combined classifier would work in each experiment, we designed stand alone baseline models that would work on image and text inputs respectively. The experiments were:

- Experiment 1: To show how the native models would behave on the dataset and what other experiments, dataset changes, etc. are needed.
- Experiment 2: Introduced normalized class weights to mitigate dataset imbalance. It showed us that class weights help but quality of extracted text was poor.
- Experiment 3: Create a sub-dataset of 24 images of each category, hand extracted text and re-ran models. We found that dataset size is too small in proportion to the

number of classes but if we have a high quality of extracted text the classification accuracy does improve.

- Experiment 4: Used only two categories, each having 75 images with manually extracted text. This experiment proved that if we have a number of images proportional to number of classification classes and if we have high quality extracted text the hypothesis does hold.

For each experiment we stated how the experiment's results affected our initial hypothesis and what was the cause of each result. By the end of the experiments, we successfully showed that our models were on their way to prove the hypothesis if expanded to a larger dataset.

Finally, it can be concluded that the text-only classifier is much more parameter efficient when compared to the image-only classifier. We say this because it is understood that images are and always have been a huge source of features to use as inputs for classification models but our text-only model, although on a very small scale, has shown that text extracted from the object in the image itself is also equally if not more feature rich.

In the future, there are other experiments that can be designed and explored that might yield different results.

- Experiments such as using an ensemble of binary or tertiary models to classify in pairs of 2-3 classes each might yield better results.

- Future scope could even include using a deep learning based text extraction tool like Clova AI Research's Deep Text Recognition (Baek *et al.* 2019). Such tools could help with automated high quality text extraction since it extracts text at all angles and colors.

- This model could also be expanded to include any other real-world image set that

contained images with text - road signs, food and beverages, apparel and fashion among others. This could provide the much needed larger dataset as well as provide a larger class diversity that the model might need to better generalize.

REFERENCES

[1] Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D. 2015. VQA: visual question answering. *CoRR* abs/1505.00468.

[2] Baek, J.; Kim, G.; Lee, J.; Park, S.; Han, D.; Yun, S.; Oh, S. J.; and Lee, H. 2019. What is wrong with scene text recognition model comparisons? dataset and model analysis. *CoRR* abs/1904.01906.

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

[4] Hendricks, L. A.; Venugopalan, S.; Rohrbach, M.; Mooney, R. J.; Saenko, K.; and Darrell, T. 2015. Deep compositional captioning: Describing novel object categories without paired training data. *CoRR* abs/1511.05284.

[5] Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

[6] Hochreiter, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6:107–116.

[7] Huang, T.-H. K.; Ferraro, F.; Mostafazadeh, N.; Misra, I.; Agrawal, A.; Devlin, J.; Girshick, R. B.; He, X.; Kohli, P.; Batra, D.; Zitnick, C. L.; Parikh, D.; Vanderwende, L.; Galley, M.; and Mitchell, M. 2016. Visual storytelling. In *HLT-NAACL*.

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

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

[10] Mikolov, T.; Grave, E.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

[11] Ni, J.; Li, J.; and McAuley, J. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 188–197. Hong Kong, China: Association for Computational Linguistics.

[12] Singh, A.; Natarajan, V.; Shah, M.; Jiang, Y.; Chen, X.; Batra, D.; Parikh, D.; and Rohrbach, M. 2019. Towards VQA models that can read. *CoRR* abs/1904.08920.