CSE3013 - Artificial Intelligence

Assistive Vision: Auto Caption and Speech Generation

Faculty: Prof. Rajeshkannan R

Team ID: 5

Slot : A1 + TA1

Team Members

1. Vanshika Nehra - 20BCE0599
2. Arya Jay Wadhwani - 20BCE0399
3. Arush Saxena - 20BCE2106
4. Oishi Poddar - 20BCE0187

1. Abstract

Visually impaired people and senior citizens are unable to identify the objects in front of them. With our project, we aim to provide assistive vision which will give a vivid description of the object in front of them and help them to identify it.
According to the recent census, it is said that 2.2 billion people suffer from visual disability all over the world, and require assistance for daily activities. Vision impairment poses an enormous global financial burden with the annual global costs of productivity losses associated with vision impairment from uncorrected myopia and presbyopia alone estimated to be US$ 244 billion and US$ 25.4 billion.We aim to provide a visual aid that improves daily performance, and independent living, thereby enhance the quality of life among these people.
Our main objective is to make an ML model that will analyze the picture that the user captures on their screen and voice out the components in it. We will be creating a text description of the captured image and then accurately convert the text into audio using Google Speech API.
Current technology allows applications to be efficiently distributed and run on mobile and handheld devices, even in cases where computational requirements are significant. Apps like; VoiceOver, Siri, Lookout, Be My Eyes, Blind Bargains etc have helped blind people do their daily activities.

2. Introduction

Motivation & Significance
We aim to provide a visual aid that improves daily performance, and independent living,thereby enhancing the quality of life among these people.
Scope and Applications:
We will be creating a text description of captured image and then accurately convert the text into audio using Google Speech API
Main Objectives of our project include:
- To develop an application to help visually impaired people gain understanding of the environment
- To generate captions that will be converted to speech thereby helping people to listen and gain clarity
- To develop a neural network model to help generate captions along with an API to convert them back to Speech
- To consider various shortcomings based on the visual context and involve all the approaches in the project

3. Literature review

Paper Title	**Algorithms used**	**Challenges**	**Observations**
Building up multi-layered perceptrons as classifier system for decision support [1]	Self-configuration algorithm and Fast Back-propagation algorithm	(1) the classification capability of multi-layered perceptrons; (2) the self-configuration algorithm for facilitating the design of the neural nets7 structure; (3) the application of the fast BP algorithm to speed up the learning procedure.	Using the self-configuration algorithm and the fast BP algorithm, a network with the structure of (33-9-3-3) is obtained and the iteration number is 5938 when convergence.
Network Compression via Mixed Precision Quantization Using a	The proposed method uses a Multi-Layer-Perceptron	Training a neural network in discrete space is challenging and the	The proposed method compresses VGG16 up to 6x when compared to the

Multi-Layer Perceptron for the Bit-Width Allocation [2]	(MLP) where the input to the MLP is the KL-divergence between the softmax output of the full precision and the quantized network and the output is the bit-width configuration for the compressed network.	convergence is slow. There are challenges of quantization too which include quantifying deviations and the many to one problem. The proposed solution consists of three main steps: sampling the search space to create a custom training set for the MLP, MLP model training, and prediction of the bit-width configuration for the compressed network	full precision model (32-bit precision weights and activations) with no accuracy drop. On ResNet50 and GoogLeNet, the method achieves up to 4x compression compared to the full precision model (32-bit precision weights and activations) while maintaining the inference accuracy. It gives better performance than uniform bit-width allocation on VGG16, ResNet50 and GoogLeNet.
Digital modulation classification using multi-layer perceptron and time-frequency features [3]	The proposed approach is based on the time-frequency features of digitally modulated signals and the use of a more robust artificial neural network, commonly referred to as an MLP.		The proposed MLP classifier with time-frequency features improves the probability of correct classification in a noisy environment.
A Reliable Localization Algorithm Based on Grid Coding and Multi-Layer Perceptron [4]	a new reliable localization algorithm, which firstly utilizes quantized RSS to encode the monitoring region which has been divided into grids, so as to specify the grids that the interested target appears roughly.	The traditional RSS-based fingerprint localization algorithm needs RSS values from all access points (AP) at each reference point (RP). In the large-scale indoor environment, the increasing of the number of APs will lead to establishing a large-scale fingerprint database, which occupies a lot of storage space.	It can be concluded from the experimental results that compared with the traditional MLP-based fingerprint localization algorithm, the proposed algorithm reduces the size of fingerprint database over 80% with guarantee of localization accuracy. Moreover, our algorithm can obtain better localization accuracy compared with the other latest quantization based localization algorithm.
Convolution in Convolution for Network in Network [5]	The proposed method is implemented by applying unshared convolution across the channel dimension and applying shared convolution across the spatial dimension in some com-putational layers. The proposed method is called	NiN utilizes shallow multilayer perceptrons (MLP). MLP itself consists of fully connected layers that give rise to a large number of parameters. There is a need for the dense shallow MLPto be replaced with sparse	The experimental results on the CIFAR10 dataset, augmented CIFAR10 dataset, and CIFAR100 data set demonstrate the effectiveness of the proposed CiC method.

	convolution in convolution (CiC)	shallow MLP.	
Efficient Convolution Neural Networks for Object Tracking Using Separable Convolution and Filter Pruning [6]	In this paper, the standard convolution is optimized by the separable convolution, which mainly includes a depth wise convolution and a pointwise convolution. To further reduce the calculation, filters in the depth wise convolution layer are pruned with filter variance.	Object tracking based on deep learning is a hot topic in computer vision with many applications. Due to high computation and memory costs, it is difficult to deploy convolutional neural networks (CNNs) for object tracking on embedded systems with limited hardware resources	With the improvements, the number of parameters is decreased to 13% of the original network and the computation is reduced to 23%. On the NVIDIA Jetson TX2, the tracking speed increased to 3.65 times on the CPU and 2.08 times on the GPU, without significant degradation of tracking performance in VOT benchmark
A Deep Neural Framework for Image Caption Generation Using GRU-Based Attention Mechanism [7]	CNN and RNN Architecture and Algorithm with attention mechanisms followed by a GRU model merged with the Bahdanau attention model on the MSCOCO dataset.	Such a model requires a high hardware overhead along with its requirements posed a challenge. Also, the image had to be pre processed before feeding it into the Neural Network and that required a different kind of segmentation knowledge and survey filtering based caption generative dataset. Forming that was a big challenge as well.	To create the descriptive sentence, a language model called GRU was chosen as the decoder. Meanwhile, combining the Bahdanau attention model with GRU allowed learning to be focused on a specific portion of the image in order to improve performance. The entire model can be fully trained using stochastic gradient descent, which simplifies the training procedure. Experiments show that the suggested model is capable of automatically generating appropriate captions for images.
Contextual LSTM for Large Scale NLP Tasks [8]	CLSTM Algorithm is trained on two wiki documents along with an additional proposed RNN LM for adding extra context. NN-HMM Model is also proposed as an alternative where each node of a CNN corresponds to a higher level feature.	Overall there are 5000 sentence sequences in the final dataset. For each sequence prefix AiBiCi, the model has to choose the best next sentence Di from the competing set of next sentences. The average accuracy of the baseline LSTM model on this dataset is 52%, while the average accuracy of the CLSTM model using word + sentence-level topic features is 63% (as	Using contextual features in a CLSTM model can be beneficial for different NLP tasks like word prediction, next sentence selection and topic prediction. For the word prediction task CLSTM improves on state-of-the-art LSTM by 2-3% on perplexity, for the next sentence selection task CLSTM improves on LSTM by ≈20% on accuracy on

		shown in Table 3). So the CLSTM model has an average improvement of 21% over the LSTM model on this dataset.	average, while for the topic prediction task CLSTM improves on state-of-the-art LSTM by ≈10% (and improves on BOWDNN by ≈7%).
A CRNN-GRU based reinforcement learning approach to audio captioning [9]	This paper proposes the use of a powerful CRNN encoder combined with a GRU decoder to tackle this multimodal task. In addition to standard cross-entropy, reinforcement learning is also investigated for generating richer and more accurate captions	The improvement in ROUGEL and METEOR is not as significant as other metrics. The improvement in ROUGEL and METEOR is not as significant as other metrics. There is only a slight difference between this submission and the submission ranking the third (0.194 / 0.196). Even though the prediction accurately describes the audio event, it is not as detailed as the human annotations. The human annotations may contain specific descriptions like “vibrating”“buzzing” while the model prediction only generates “running”. Due to the limited information in audio as well as the direct optimization towards CIDEr metric, the model chooses to output a correct, yet general description of the audio events.	A novel audio captioning approach utilizing a CRNN encoder front-end as well as a reinforcement learning framework. Audio captioning models are trained on the Clotho dataset. The results on the Clotho evaluation set suggest that the CRNN encoder is crucial to extract useful audio embeddings for captioning while reinforcement learning further improves the performance significantly in terms of all metrics
Adversarial Robust Transfer Learning [10]	By training classifiers on top of these feature extractors, we produce new models that inherit the robustness of their parent networks. We then consider the case of “fine tuning” a network by re-training end-to-end in the target domain.	When the goal is to produce a model that is not only accurate but also adversarially robust, data scarcity and computational limitations become even more cumbersome.	By using such strategies, it is possible to produce accurate and robust models with little data, and without the cost of adversarial training.

4. Problem Statement and Description

Visually impaired people and senior citizens are unable to identify the objects in front of them. According to the recent census, it is said that 2.2 billion people suffer from visual disability all over the world, and require assistance for daily activities.

Vision impairment poses an enormous global financial burden with the annual global costs of productivity losses associated with vision impairment from uncorrected myopia and presbyopia alone estimated to be US$ 244 billion and US$ 25.4 billion.We aim to provide a visual aid that improves daily performance, and independent living, thereby enhance the quality of life among these people. The 2.2 million people that suffer from visual disability; we can say that the need for its aid is high in demand and will increase as time goes by and as technology expands.

Current technology allows applications to be efficiently distributed and run on mobile and handheld devices, even in cases where computational requirements are significant. Apps like; VoiceOver, Siri, Lookout, Be My Eyes, Blind Bargains etc have helped blind people do their daily activities. Hence with our project, we aim to provide assistive vision which will give a vivid description of the object in front of them and help them to identify it.

The current technology does not provide any assistance apart from hearing aids or strong lenses for people that are visually impaired. For people that cannot afford huge surgeries like Laser surgeries find it very difficult to get accustomed to the surroundings. A National Geographic Survey conducted in 2016 posed a serious threat to the vision of ordinary people. An increase in digital technology has increased the screen time of huge masses and an increase in visual impairment as well.

There is no product on the market that can help provide a basic level of assistance for people that cannot afford big equipment and surgeries. Here is where our product helps the masses without paying a single penny. Our product is in the form of an application that will help people get a sense of their surroundings and help them in doing normal routine based work with the help of our product.

Our main objective is to make an ML model that will analyze the picture that the user captures on their screen and voice out the components in it. We will be creating a text description of the captured image and then accurately convert the text into audio using Google Speech API.

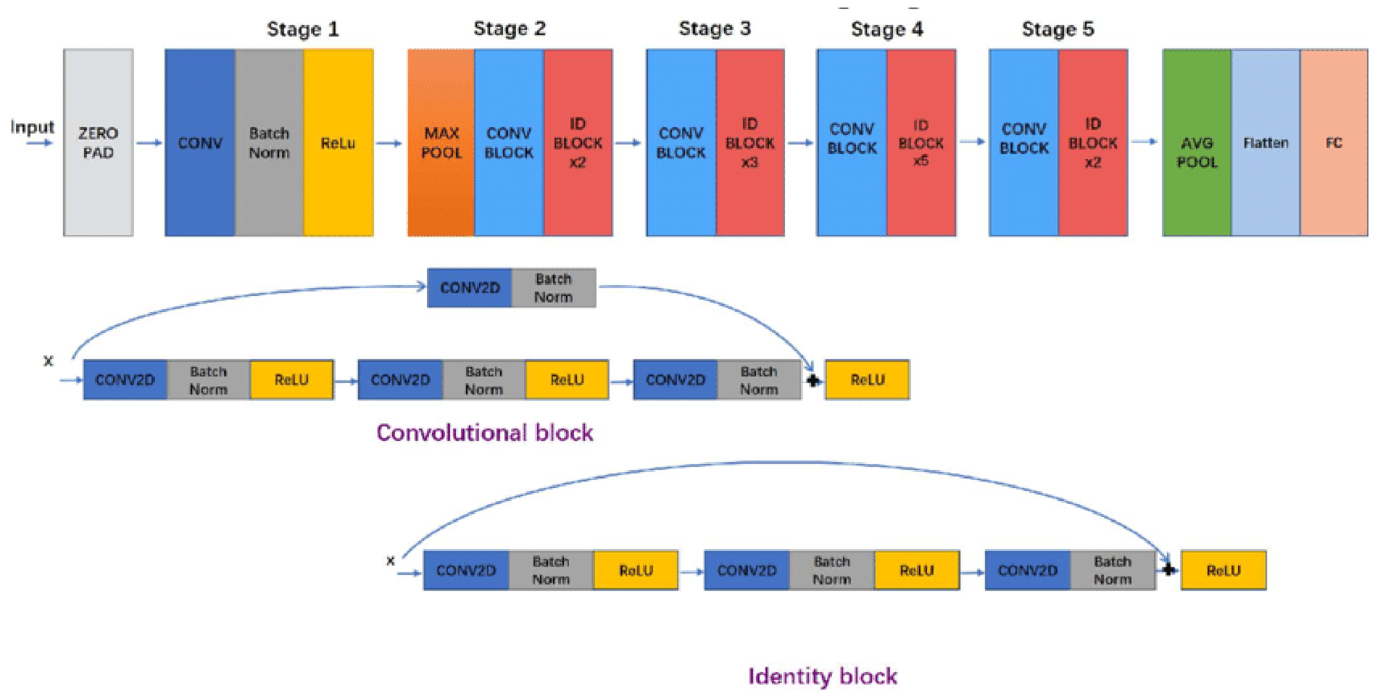
Main Objectives of our project include:

- To develop an application to help visually impaired people gain understanding of the environment
- To generate captions that will be converted to speech thereby helping people to listen and gain clarity
- To develop a neural network model to help generate captions along with an API to convert them back to Speech
- To consider various shortcomings based on the visual context and involve all the approaches in the project

In our project, The dataset is a sentence-based image description and search, consisting of 8,000 images that are each paired with five different captions which provide clear descriptions of the salient entities and events. The images are chosen from six different Flickr groups, and tend not to contain any well-known people or locations, but are manually selected to depict a variety of scenes and situations. OpenCv is used for preprocessing of images, CNN has been used for model creation, ResNet50 is used for training the model, Captions are generated using LSTM and Voice is generated using gTTs.

This project will help provide an easy yet efficient solution to people that are visually impaired and face difficulties in performing basic work. Just a click from an app will help them get audio based on the context obtained from the image with the help of the various ML models and Neural Networks as mentioned above.

4.1 Architecture Diagram



The above diagram shows the RESNET50 Architecture and how the various layers are placed in various blocks. ResNet50 is a variant of the ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations. Given below is the layer wise stride and filter that we will be applying in the various layers.

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

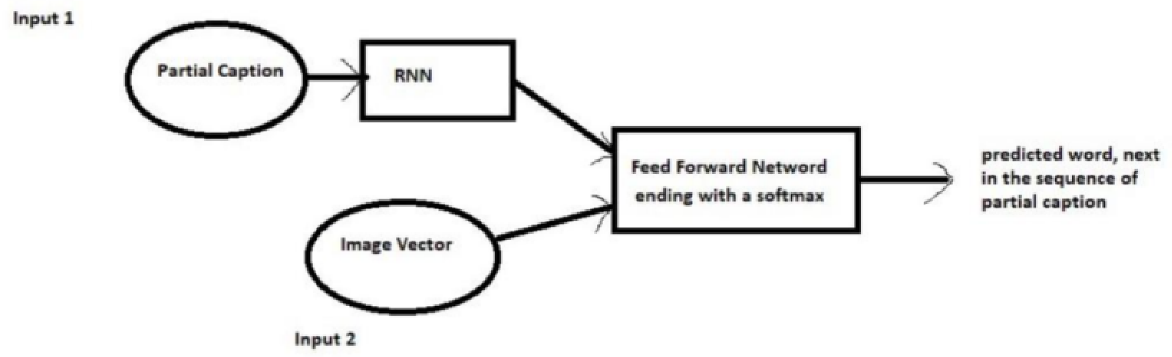
All the layers as presented in the above table can be broken down as:

- A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us 1 layer.
- Next we see max pooling with also a stride size of 2.

- In the next convolution there is a 1 * 1,64 kernel following this a 3 * 3,64 kernel and at last a 1 * 1,256 kernel, These three layers are repeated a total 3 times so giving us 9 layers in this step.
- Next we see kernel of 1 * 1,128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512 this step was repeated 4 times, giving us 12 layers in this step.
- After that there is a kernel of 1 * 1,256 and two more kernels with 3 * 3,256 and 1 * 1,1024 and this is repeated 6 times giving us a total of 18 layers.
- And then again a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048 and this was repeated 3 times giving us a total of 9 layers.
- After that we do an average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

We don't actually count the activation functions and the max/ average pooling layers so totaling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network.

4.2 Flow Diagram


Input 1
Partial Caption
RNN
Feed Forward Netword ending with a softmax
predicted word, next in the sequence of partial caption
Image Vector
Input 2

The above diagram shows how the inputs (images and captions) are taken through the neural network. Here we use the RNN architecture for the Captions whereas the CNN architecture for the Images that are fed into a feed forward neural network with softmax function for its hyperparameter tuning.

4.3 Pseudocode

```
import [keras,tensorflow,pandas,numpy,nltk,string,json,cv2]

function read_text_split(file) returns the captions from file
    captions={}
    open(file) and do for each Ci
        captions = captions U Ci
    for each Ci in captions do
        captions.split("\n","##")
    return captions

function assign_descriptions(captions) returns a dictionary
    descriptions ={}
    for each Ci in captions do
        split(Ci) into first,second
        if (descriptions doesn’t have first)
            descriptions[first].append(second)
    return descriptions

function clean_text(sentence) returns a clean text
    new_sentence = remove_punctuation(sentence) and remove_single_letters(sentence)
    return new_sentence

function clean_descriptions(descriptions) returns clean and refined descriptions
    key={}
    caption={}
    for each Ci,Cj in description do
        temp_caption=clean_text(Cj)
        assign(caption,i,temp_caption)
    return caption

function prepare_train_test(train, descriptions) returns training_captions
    temp={}
    train_descriptions={}
    for each id in train do
        for each Ci in descriptions at key:id do
            temp=”startseq”+temp+”endseq”
            assign(traim_descriptions,id) and append(temp)
    return train_descriptions

function define_model_resnet(50,weights,input_shape) returns a defined model
    model={}
    model=resnet50(weights,input_shape)
    refined_model = Model(model.input,model.layers.output)
    return [model,refined_model]

function pre_process_image(image,target_size) returns preprocessed image
    image = load_image(target_size)
    image = convert_to_array(image)
    image = expand_dimensions(axis=0) and normalize(image)
    return image

function encode_image(image) returns extracted feature from Resnet50
    feature_vector={}
    feature_vector = get_model from define_model_resnt50 then
        model_new.predict(image)
    return feature_vector

function encode_trainingset(train)
    encoded_train = {}
    for each image,id in train:
        assign(encoded_train,id,encode_image(image))
        return encoded_train
# repeat for test-data as well

function data_generator(train_descriptions,encoding_train,word_to_idx,max_len,batch_size)
        while true
                for  key,list in train_descriptions do
                        for desc in descriptions do
                                for i in range(1,len(seq))
                                        add padding(word)
                                        append(photo,word)
        return word

function word_embeddings returns embedding_index
        embedding_index={}
        for line in file “glove” do
                values=spit(line)
                word_embedding = embed(values)
                assign(embedding_index,word,word_embedding)
        return embedding_index

function get_embedding_matrix(embedding_index) returns embedding_matrix
        embedding_matrix = {0}
        for word,id in words do
                embedding_vector = embedding_index[word]
                assign(matrix,id,embedding_vector)

        return matrix

function define_model(img) returns model embedded with neural network layers
        input_img_features = Input(shape=(2048,))
        inp_img1 = Dropout(0.3)(input_img_features)
        inp_img2 = Dense(256,activation='relu')(inp_img1)
        input_captions = Input(shape=(max_len,))
        inp_cap1 = Embedding(input_dim=vocab_size,output_dim=50,mask_zero=True)(input_captions)
        inp_cap2 = Dropout(0.3)(inp_cap1)
        inp_cap3 = LSTM(256)(inp_cap2)
        decoder1 = add([inp_img2,inp_cap3])
        decoder2 = Dense(256,activation='relu')(decoder1)
        outputs = Dense(vocab_size,activation='softmax')(decoder2)
        model = Model(inputs=[input_img_features,input_captions],outputs=outputs)

        return model

function train_model(train,epochs,batch_size,steps, train_descriptions,max_len,verbose) returns model
        for i in range(epochs) do
                generator = data_generator(train_descriptions, encoding_train,words,max_len,batch_size)
                model.fit_generator(generator,epochs,steps,verbose)
                save(model)
                compile(model,optimizer=’sgd’,loss=’mse’,metrics)
        return model

captions=predict(photo) from model
audio=GoogleTextToSpeech(captions)
```

5. Experimentation and Results

5.1 Data Set (Sample with Explanation)

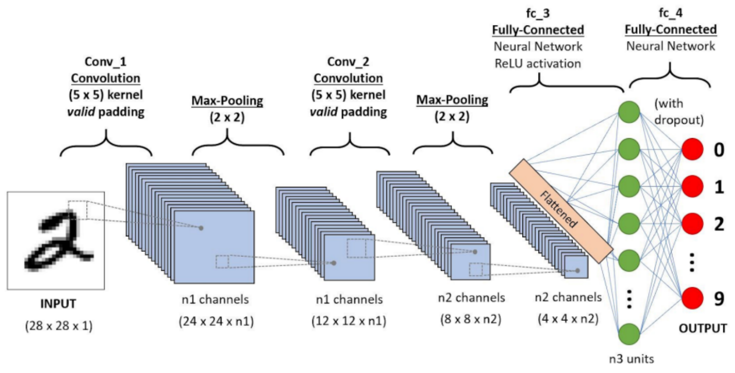
LINK: Flickr 8k Dataset | Kaggle
This dataset has around 8000 images along with its captions. Every image has a label and corresponding to that, we have its captions in the captions.txt file given.
For example, the first image labeled "1000268201_693b08cb0e.jpg" is an image that has the caption as "A child in a pink dress is climbing up a set of stairs in an entryway." The image that was given is shown below:

The images used for testing must be semantically related to those used for training the model. For example, if we train our model on the images of cats, dogs, etc. we must not test it on images of air planes, waterfalls, etc. This is based on the fact that the model puts forward its predictions based on the training dataset. The model is able to identify the colors precisely as well.

5.1.1 Explain methodology with the dataset

The two Neural Networks that will be used in this project are explained below and using the images, we will be explaining it in detail.

CNN - Convolutional Neural Network(ResNet50)



Above given is an example on how images are processed under the CNN architecture. The first part is what we call the kernel padding which targets a particular feature of the image. In our case, it will be the main subject of the image. Like in the example mentioned in **Img5.1(1)** we will first extract the features, like the girl based on the padding, a small pixelated area which we will expand in channels which is called pooling.

The pooling will further target a minute feature and further divide them again using kernel padding and this way multiple features of the images are analyzed after which all are flattened(converted into a single dimension vector) which are fed into the dense layer. The dense layer then compacts them and gives us a classification based output. Combining this with the RNN for captions, we will be having a full fledged Model that will have captions generated from the images.
The layers in this stage help in analyzing the input images via a ResNet50 Model, that categorizes images via gradients, edges and color in the starting layers and the further layers train to understand shapes like wheels and people. The Model targets various features in this way, further strengthening its accuracy.

We will be using the Flickr Dataset which has images and captions embedded in them. Kaggle will present this dataset which will be loaded on Google Colab using Pandas as the library. First we performed data preprocessing which involved converting the captions into proper generated sentences with all quotations and unnecessary words removed using the basic split function. We further removed the stopwords and found the keywords using regex and frequency counter of words that give us the most important partially generated caption. We then use the OpenCV library in Python to read the image that is given in the dataset.

ResNET50 model is used here with the Input Layer of the shape (None,224,224,3) connected to the Zero Padding convolution layer with output shape (None,230, 230,3) passed onto the conv2D layer(None,112,112,64) with Batch Normalization as well. A relu activation function is used in this layer which is then passed onto the 2 2D pooling, one with zero padding with an output shape of (None,114,114,64) and (None,56,56,56). We then pass onto the 2nd Convolution layer with various normalization and activation functions such as relu and out with Batch Normalization. 5 Layers like these are repeated with various output shapes based on the RGB configurations. We finally put the predictions onto a Dense Layer before which GlobalAveragePooling was performed.

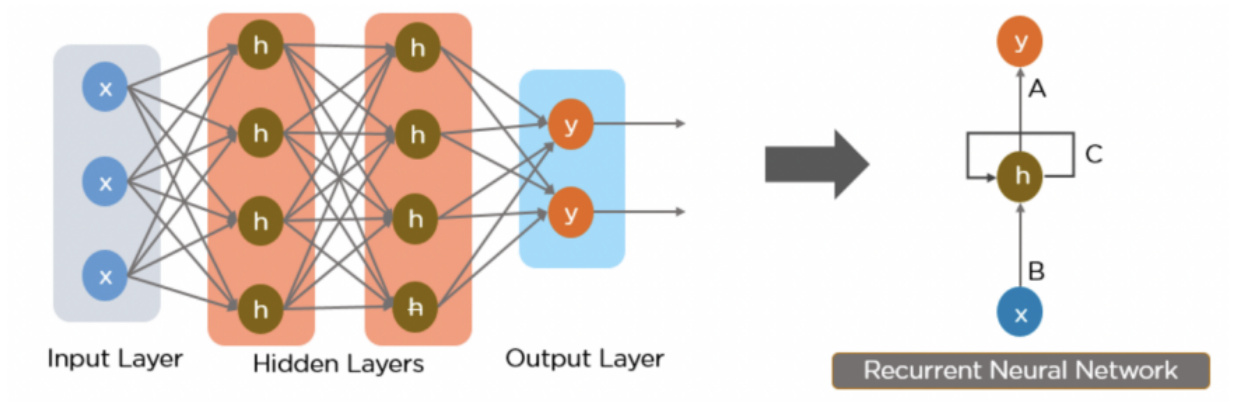
The layers that will be used in the ResNet50 model are:

- A convolution with a kernel size of 7 * 7 and 64 different kernels all with a stride of size 2 giving us 1 layer.
- Next we see max pooling with also a stride size of 2.
- In the next convolution there is a 1 * 1,64 kernel following this a 3 * 3,64 kernel and at last a 1 * 1,256 kernel, These three layers are repeated a total 3 times so giving us 9 layers in this step.

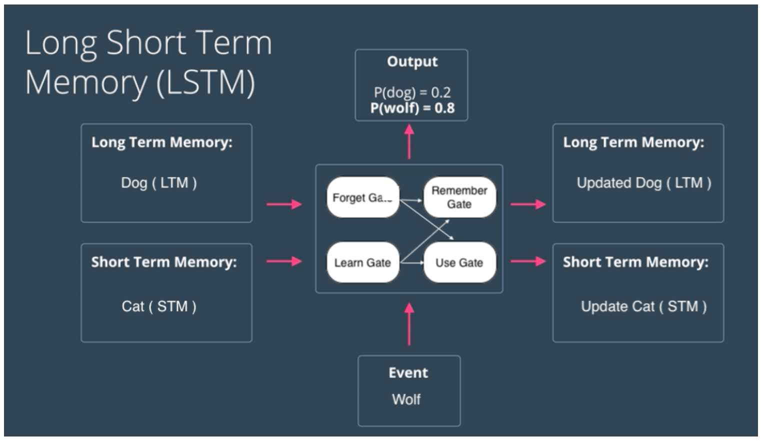
- Next we see kernel of 1 * 1,128 after that a kernel of 3 * 3,128 and at last a kernel of 1 * 1,512 this step was repeated 4 times, giving us 12 layers in this step.
- After that there is a kernel of 1 * 1,256 and two more kernels with 3 * 3,256 and 1 * 1,1024 and this is repeated 6 times giving us a total of 18 layers.
- And then again a 1 * 1,512 kernel with two more of 3 * 3,512 and 1 * 1,2048 and this was repeated 3 times giving us a total of 9 layers.
- After that we do an average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us 1 layer.

Using these predictions, we again convert the predictions back into the sentences and use the Google Text to Speech API as well.

LSTM - LongShort Term Memory



Unlike the CNN where we have pre defined layers, here in the RNN we have Hidden Layers that help analyze the various categorical features and based on multiple input features, categorizes them in either x or y. This helps in using the features extracted from the images and feeding the, in this RNN that will give partial captions being generated transforming into full statements using the LSTM model. Features like, a girl and the stairs will be taken as input using the feature extraction method explained above and will be fed into the Neural Network and will keep iterating inside the hidden layers till it gives a conclusion as predicted words.

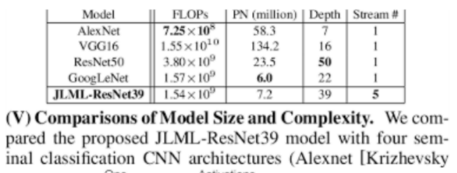


At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function. In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Input gate is used to quantify the importance of the new information carried by the input. So based on the current expectation we have to give a relevant word to fill in the blank. That word is our output and this is the function of our Output gate.

Complexity Analysis

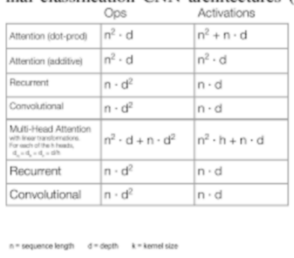
For a method to be working and efficient, analyzing the complexity and overhead right from the start is essential. Here is a table that reflects the various analogies pertaining to the Algorithms and Neural Networks being used:

CNN and RNN:



Model	FLOPs	PN (million)	Depth	Stream #
AlexNet	$\mathbf{7.25 \times 10^8}$	58.3	7	1
VGG16	1.55×10^{10}	134.2	16	1
ResNet50	3.80×10^9	23.5	**50**	1
GoogLeNet	1.57×10^9	**6.0**	22	1
JLML-ResNet39	1.54×10^9	7.2	39	**5**

(V) Comparisons of Model Size and Complexity. We compared the proposed JLML-ResNet39 model with four seminal classification CNN architectures (Alexnet [Krizhevsky



	Ops	Activations
Attention (dot-prod)	$n^2 \cdot d$	$n^2 + n \cdot d$
Attention (additive)	$n^2 \cdot d$	$n^2 \cdot d$
Recurrent	$n \cdot d^2$	$n \cdot d$
Convolutional	$n \cdot d^2$	$n \cdot d$
Multi-Head Attention with linear transformations. For each of the h heads, $d_q = d_k = d_v = d/h$	$n^2 \cdot d + n \cdot d^2$	$n^2 \cdot h + n \cdot d$
Recurrent	$n \cdot d^2$	$n \cdot d$
Convolutional	$n \cdot d^2$	$n \cdot d$

n = sequence length d = depth k = kernel size

Above given is the activation function complexity as well as the operations that will be performed. We can see that RNN and CNN have a much lesser activation function complexity as compared to others and hence, it acts as a viable option to be used in this project.

6. Conclusion

Our main objective of this project was to create a visual assistance for the visually impaired which will give them a description of the features of the object in front of them. With the help of our ML model we have achieved this objective by extracting the features of the picture in the dataset and storing it in a pickel file, followed by caption generation and voice generation.

7. Future Work

When we improve the program the accuracy and efficiency of the model will increase and we can auto play the audio and let the user click a picture via their device and even connect their contacts to their families for face recognition and other AI features. This can be used for other social media accounts where the user doesn't need to click a picture but the AI just does an immersive reader not only of the text but also of the pictures that are displayed on the screen. Improvements can be made as much as possible for the model and testing and training using large datasets to improve accuracy can be done. When spoken about accuracy of the model, there are many minute details to be taken care of and when the image dataset and vocabulary expands such details can be noticed and can be rectified. A lot of modifications can be made to improve this solution like: a lot of modifications can be made to improve this solution like:

•Using a larger dataset
•Changing the model architecture, e.g. include an attention module.
•Doing more hyper parameter tuning (learning rate, batch size, number of layers, number of units, dropout rate, batch normalization etc.).
•Use the cross validation set to understand overfitting.
•Using Beam Search instead of Greedy Search during Inference.
•Using BLEU Score to evaluate and measure the performance of the model.

Colab Link:
https://colab.research.google.com/drive/19JU7nm6zTr0NK9q4wOTQPDyVBHJYtuKD?authuser=1

8. References

[1] Jun, C., Fan, Z., & Shan, F. (1995). Building up multi-layered perceptrons as classifier system for decision support. *Journal of Systems Engineering and Electronics*, *6*(2), 32-39.

[2] Soufleri, E., & Roy, K. (2021). Network Compression via Mixed Precision Quantization Using a Multi-Layer Perceptron for the Bit-Width Allocation. *IEEE Access*, *9*, 135059-135068.

[3] Ye, Y., & Wenbo, M. (2007). Digital modulation classification using multi-layer perceptron and time-frequency features. *Journal of Systems Engineering and Electronics*, *18*(2), 249-254.

[4] Sun, Z., Zhang, Y., & Ren, Q. (2020). A reliable localization algorithm based on grid coding and multi-layer perceptron. *IEEE Access*, *8*, 60979-60989.

[5] Pang, Y., Sun, M., Jiang, X., & Li, X. (2017). Convolution in convolution for network in network. *IEEE transactions on neural networks and learning systems*, *29*(5), 1587-1597.

[6] Mao, Y., He, Z., Ma, Z., Tang, X., & Wang, Z. (2019). Efficient convolution neural networks for object tracking using separable convolution and filter pruning. *IEEE Access*, *7*, 106466-106474.

[7] Khan, R., Islam, M. S., Kanwal, K., Iqbal, M., Hossain, M., & Ye, Z. (2022). A Deep Neural Framework for Image Caption Generation Using GRU-Based Attention Mechanism. *arXiv preprint arXiv:2203.01594*.

[8] Ghosh, S., Vinyals, O., Strope, B., Roy, S., Dean, T., Heck, L., & Contextual, L. S. T. M. (2016). models for large scale NLP tasks. *arXiv preprint arXiv:1602.06291*.

[9] Xu, X., Dinkel, H., Wu, M., & Yu, K. (2020, November). A crnn-gru based reinforcement learning approach to audio captioning. In *Proceedings of the Detection and Classification of Acoustic Scenes and Events Workshop (DCASE)* (pp. 225-229).

[10] Shafahi, A., Saadatpanah, P., Zhu, C., Ghiasi, A., Studer, C., Jacobs, D., & Goldstein, T. (2019). Adversarially robust transfer learning. *arXiv preprint arXiv:1905.08232*.