Machine Learning Engineer Nanodegree

Capstone Project

Visual Question Answering System

Abishek Chiffon M,
June 14th,2019

Domain Background

Deep learning is the field which of computer science which has stolen the world's attention over the last decade. Even though most of the algorithms of deep learning were invented in the early 2000s, the algorithms came to be used only after the GPU industry's phenomenal growth during 2012. Since then Deep learning Neural Networks like Convolutional neural network and Recurrent Neural Network has solved many Computer vision and Natural Language processing problems.

Natural Language Processing allows computers to understand human's first invention- Language. NLP has always baffled the research community because of the multiple ways the same text can be interpreted. Inventions like attention followed by Transformers, BERT has made phenomenal improvements in Neural Machine Translation and Sentiment analysis.

However, the current state of the art demonstrates that a coarse scene-level understanding of an image paired with word n-gram statistics suffices to generate reasonable image captions, which suggests image captioning may not be as "AI-complete" as desired.
What makes for a compelling "AI-complete" task? We believe that in order to spawn the next generation of AI algorithms, an ideal task should require multi-modal knowledge beyond a single sub-domain (such as CV)

Since the convolutional neural networks with images perform better than RNN's in language tasks, using images in the language tasks will lead to better language

understanding models like the Listen and Watch pedagogy improves children's memory retention.

Problem statement

The goal of the project is to help Blind people to receive answers to questions about their surroundings. Instead of always depending on a person to explain their surroundings, Blind people can take a photo of it and let the phone answer them with an yes or no. The problem I have decided to tackle is Visual Question Answering System.
Given an image and a natural language question about the image, the task is to provide an accurate Positive or negative answer.[4]

In VQA Dataset from www.visualqa.org, the computer system needs to address issues, such as, a binary classification problem (Is the umbrella upside down?), a counting problem (How many children are in the bed?), or an open-ended question (Who is wearing glasses? Where is the child setting?)

The Authors of [2] and [4] has done considerable amount of data collection and data verification on the VQA to give us something to work on.

Visual questions selectively target different areas of an image, including background details and underlying context.

Datasets and Inputs

VQA is a new dataset containing open-ended questions about images.
These questions require an understanding of vision, language and commonsense knowledge to answer.

1. 80k(COCO and abstract scenes)
2. At least 3 questions (5.4 questions on average) per image
3. The class labels are "Yes" and "No"

Solution Statement

This is a supervised classification encoder-decoder model. Where the images and questions are inputs, while the outputs are the answers.

The images are passed through a InceptionV3 model and the features are extracted. The questions are passed through a ElMo[3] embedding to get the word vectors. The word vectors helps our word tokens to understand the similarity between them. The ones closer in meaning has very high cosine similarity. These word vectors are then passes through two LSTM layers. The classification is done based on the softmax predictions.

Evaluation metric

A model is graded based on the accuracy metric between the predicted probability and the observed target measured on the test data. Before comparison, all responses are made lowercase, numbers converted to digits, and punctuation & articles removed. We avoid using soft metrics such as Word2Vec, since they often group together words that we wish to distinguish, such as “left” and “right”. We also avoid using evaluation metrics from machine translation such as BLEU and ROUGE because such metrics are typically applicable and reliable for sentences containing multiple words.
In VQA, most answers (89.32%) are a single word; thus there no high-order n-gram matches between predicted answers and ground-truth answers, and low-order n-gram matches degenerate to exact-string matching. Moreover, these automatic metrics such as BLEU and ROUGE have been found to poorly correlate with human judgement for tasks such as image caption evaluation.Accuracy is not a perfect measure when it comes to imbalanced dataset and ROC could tell more about the consistency of the model. But as there are few benchmarks to the model and the one I considered used accuracy.
So I decided to pseudo-balance the dataset using penalized classification. Penalized classification imposes an additional cost on the model for making classification mistakes on the minority class during training. These penalties can bias the model to pay more attention to the minority class.
I used class weights of 0.85 to the ‘no’ answer and 1 to the ‘yes’ answer, So the loss values of the ‘no’ answer will be lesser than that of the “yes”. In other words you assign higher values to the loss from the “yes” compared to that of the “no” label while computing the loss value.So in the event of misclassification because of unbalanced data, the weights of the class stabilizes them.
With this I believed that I can use accuracy safely to compare with other algorithms I considered. But on any other algorithms I would have picked F

II. Analysis

The dataset I chose had almost 200k images and 400k questions and answers. The problem was to answer the “yes/no” questions of the images. So I preprocessed and trimmed the dataset to contain 80k images. I believed this much data will be enough to answer “yes/no” questions. I chose only the questions with a maximum length of 30. The total number of samples available in the dataset is

	Total Records
The total no of answers	95163
The total no of images	95163
The total no of questions	95163

The image names are datatype string, but the original images are of type float. The questions and answers are of string type.
The questions like “Is the. . . ”, “Are. . . ”, and “Does. . . ” are typically answered using “yes” and “no” as answers. Other questions such as “What is. . . ” and “What type. . . ” have a rich diversity of responses. Other question types such as “What color. . . ” or “Which. . . ” have more specialized responses, such as colors, or “left” and “right

The images are read in RGB scale and the average pixel intensities are found to be

```
The RGB mean values for No answers                [117.71135342 112.0535665  102.52564799]
The RGB mean values for Yes answers               [118.7029844  112.96209297 103.72230445]
The RGB std values for No answers                 [32.91562039 33.44795909 38.87515253]
The RGB std values for Yes answers                [31.72024538 31.7075329  36.72497776]
```

Exploratory visualizations:

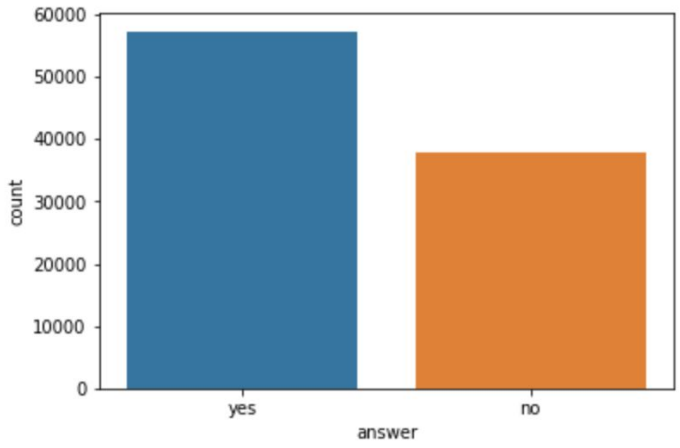
5 Random questions from the sample and their respective answers.

question	answer
is this during the summer	yes
is the tennis player jumping in the air	yes
is the traffic light attached to the building	no
is there a baby in this picture	yes
are the microwave new	yes



The above are the respective images from which the questions were asked.

The distribution of the different type of answers(classes). The distributions is not very well balanced but inorder to compare with the benchmark model, we will use accuracy measure over F2 score. Inorder to counter the imbalance in data I set the class weights to 1:0.85 after few steps. This would avoid overfitting the yes answer



Algorithms and Techniques

Visual question answering is a multimodal solution to language understanding. Multimodal solutions are hard to train. Few years ago the NLP community used a classical solution like SVM and logistic regression to solve the NLP problems. Even though they had a lot of data, they couldn't train the machine to understand it until Deep learning came into the picture . So I tried only the deep learning techniques to solve the problem. As this problem involves two input sources image and the language, I thought to build an attention model between image and language.

As a base model I planned to implement using multilayer perceptron
The disadvantage of multilayer perceptron is vanishing gradient. And there is no temporal information reserved for the language.

The final model is simpler than the attention model in which I combine the InceptionV3 features with LSTM layers to preserve the temporal information. I chose this model because of its simple and elegant solutions to other NLP problems.

Benchmark model

An LSTM with two hidden layers is used to obtain 2048-dim embedding for the question. The embedding obtained from the LSTM is a concatenation of last cell state

and last hidden state representations (each being 512-dim) from each of the two hidden layers of the LSTM. Hence 2 (hidden layers) x 2 (cell state and hidden state) x 512 (dimensionality of each of the cell states, as well as hidden states) . This is followed by a fully-connected layer + tanh non-linearity to transform 2048-dim embedding to 1024-dim.
It performs with an accuracy of 50.39.

Methodology

Data Preprocessing

Text preprocessing

The images, questions and answers are pulled from the json and stored in a file. The questions contained unnecessary symbols which were removed by using regular expressions. The questions are tokenized using Keras tokenizer, as the texts are of different size, they are padded. While tokenizing only the top 5000 words are chosen rest of them are stored as "<unk>". We used ElMo,Rather than a dictionary of words and their corresponding vectors, ELMo analyses words within the context that they are used. It is also character based, allowing the model to form representations of out-of-vocabulary words.

This therefore means that the way ELMo is used is quite different to word2vec or fastText. Rather than having a dictionary 'look-up' of words and their corresponding vectors, ELMo instead creates vectors on-the-fly by passing text through the deep learning model.

The labels in our case are "yes" and "no". They were also tokenized.

Image preprocessing for Train data

Images were decoded and resized to 299,299, so that they could be passed through the InceptionV3 model to get the bottle-neck features.

The image data we have is about 40GB, as this is a lot of data to have in the memory it is advised to load it from disk. But I decided to sacrifice disk space for training time,

which is crucial. I constructed a tf dataset and loaded the data, extracted the InceptionV3 features from them and stored in the sample place as numpy files.

By doing this the training time will be reduced significantly.

Now the training data and test data are split in the ratio 0.8:0.2. The image,question and answer are zipped as a tf.dataset. They are shuffled and batched into size of 64.

Implementation

I tried with different algorithms and architectures to identify the best one.

Multi layer perceptron :

MLP is a simple feedforward neural net that maps a feature vector (of fixed length) to an appropriate output. In our problem, this output will be a probability distribution over the set of possible answers.

For the question, we transform each word to its word vector, and sum up all the vectors. The length of this feature vector will be same as the length of a single word vector, and the word vectors (also called embeddings) that we use have a length of 300.

For the image, we pass it through a Deep Convolutional Neural Network (the well-known Inception Architecture), and extract the activation from the second last layer (before the softmax layer, that is). Size of this feature vector is 4096.

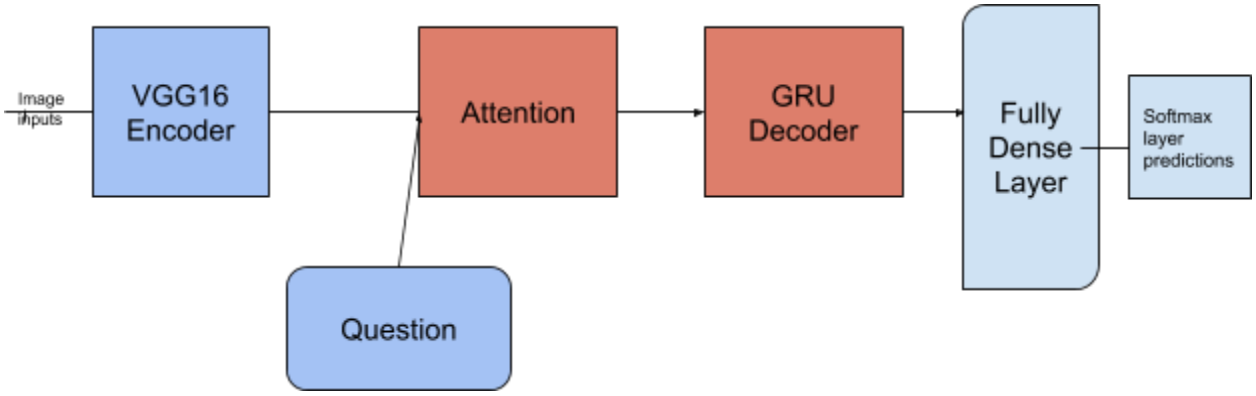
Once we have generated the feature vectors, all we need to do now is to define a model in Keras, set up a cost function and an optimizer, and we're good to go. The following Keras code defines a multi-layer perceptron with two hidden layers, 1024 hidden units in each layer and dropout layers in the middle for regularization. The final layer is a softmax layer and is responsible for generating the probability distribution over the set of possible answers. I have used the categorical_crossentropy loss function since it is a multi-class classification problem.

I was able to achieve a training speed of 400 seconds/epoch with an accuracy of about 44.89%.

Attention Model

With the motivation got from the previous model I decided to use custom training and subclassing of Tensorflow 2.0 for the attention model.

The images are passed through a Inception model and the features are extracted. These features are then passed through an attention model to focus on the areas of the image relative to the question. The attention model output and the question are then fed into a Gated Recurrent Unit.



The attention model takes in features of the images and passes it through an encoder. Then the encoder result, previous hidden state of the LSTM are sent to the attention layer. Where it finds the context vector for the question's single token. The context understands positional information of the text.

The question's token is sent to the Decoder which is then forwarded to attention layer for image attention and context vector. The context vector is concatenated with the input question token vector, they are then passed through an LSTM layer.

The LSTM's output is passed through a Dense layer. The hidden state from the LSTM is sent to the next Decoder's input.

I used the categorical loss for the loss measure, Adam as the optimizer, trained for 10 epochs over 3.5 hours.

Below is the attention and context vector formulas.

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]} \quad (1)$$

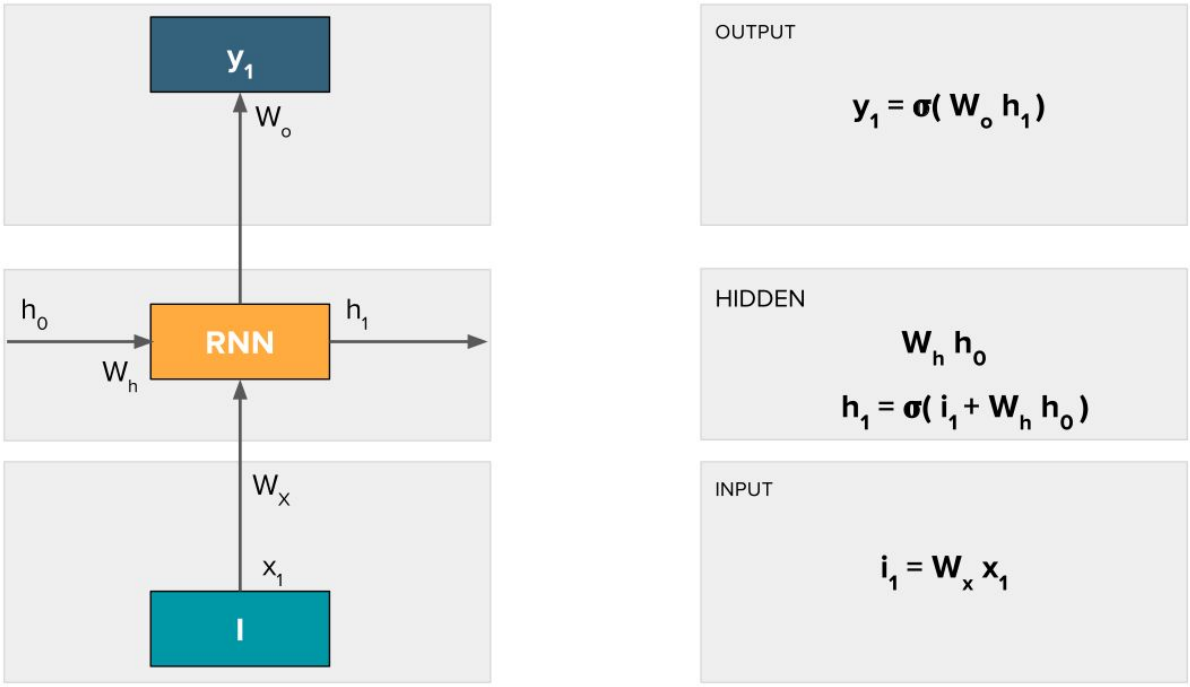
$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]} \quad (2)$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_{\boldsymbol{c}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]} \quad (3)$$

The model performed very poorly due to various reasons. The maximum accuracy was 24%.

Bidirectional LSTM model.

Having learnt from the previous model that model with over complexity goes out of hand very easily, I decided to do moderately complex Bidirectional LSTM with Functional API instead of Imperative-subclassed API.



RNN

The LSTM model is are RNN networks which takes your input and the hidden state at the a particular time to calculate current word and the next hidden state .

So in our figure the input x1 is multiplied with w1 to get the INPUT. Our algorithm calculates the W1 . The next hidden state is calculated the previous hidden state multiplied with w2 to and added with our INPUT. Then activation sigmoid is applied to constraint it between 0 and 1 . This hidden state is then sent to the next time step to calculate the its parameters. The hidden state h1 is also used to calculate our output y1 and applying sigmoid to it.

So this is the once atomic version of RNN. The LSTM has gate architecure which is used to filter the amount information in the each RNN cell. The four gate is LSTM are

LSTM

LSTM will have memory cells in addition to the hidden state to store past input's infomation. And the gates are used to arbitrate how much should the cell pass its information to the next time step.

$i(t) = \sigma(W(i)\, x(t) + U(i)\, h(t-1))$ (Input gate)

$f(t) = \sigma(W(f)\, x(t) + U(f)\, h(t-1))$ (Forget gate)

$o(t) = \sigma(W(o)\, x(t) + U(o)\, h(t-1))$ (Output/Exposure gate)

$\tilde{c}(t) = \tanh(W(c)\, x(t) + U(c)\, h(t-1))$ (New memory cell)

$c(t) = f(t) \circ c(t-1) + i(t) \circ \tilde{c}(t)$ (Final memory cell)

$h(t) = o(t) \circ \tanh(c(t))$

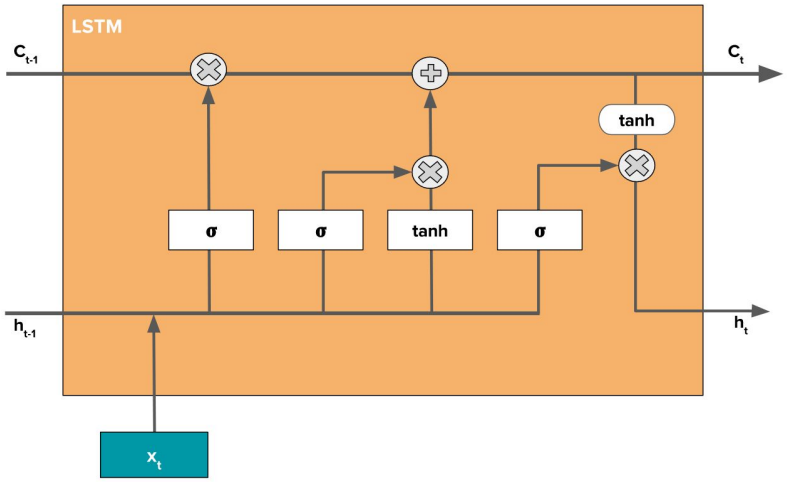
The input gate determines the current cell should be taken in to consideration while calculating the next hidden state. Except here it is not the output of the current time step.

It s just like an RNN cell's hidden state calculation.The parameters to learn are W(i) and U(i)

The forget gate decides how much of the past it should remember.

The new memory cell is the same as RNN's hidden state. In the final memory cell the amount of information to be transmitted is calculated by taking a sort of weighted percentage calculation between the current memory cell and the previous memory cell. The weights are the Input and forget gate.

Tanh activation is applied to this New memory cell along with another gate called exposure gate to determine how much of the past information should be exposed to the current step



Bidirectional LSTM

This Bidirectional LSTM has 2 passes — forward pass and backward pass.

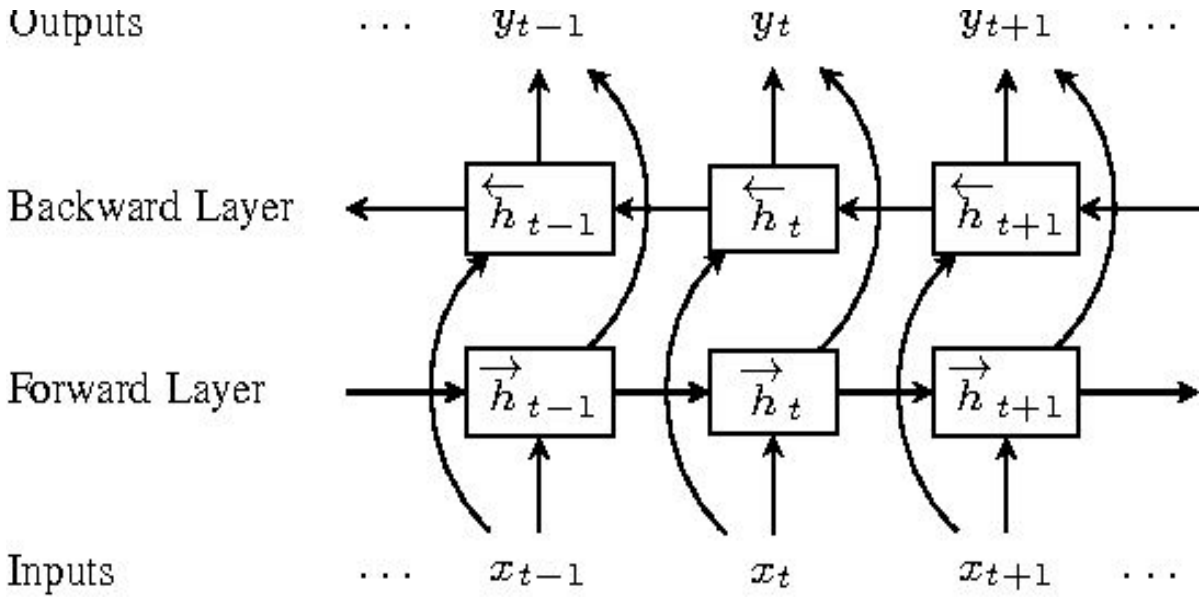
Where the information from the reverse pass is concatenated with the information from the forward pass to get the whole context of the text under consideration.

$h_1(t) = f(\overrightarrow{W} x(t) + V h(t-1) + b)$ (1)

$h_2(t) = f(\overleftarrow{W} x(t) + V h(t+1) + b)$ (2)

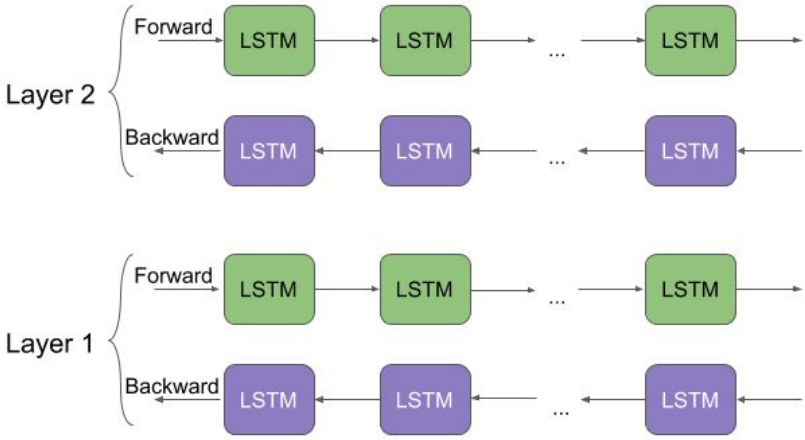
$y(t) = g(Uh(t) + c) = g(U[h_1(t) ; h_2(t)] + c)$

The h1 and h2 are the hidden states achieved from forward and backward passes of the text. The output y(t) is calculated a activation function to the concatenation of both the hidden states.



ELMO

As I mentioned earlier, ELMo word vectors are computed on top of a two-layer bidirectional language model (biLM). This biLM model has two layers stacked together. Each layer has 2 passes — forward pass and backward pass:



The architecture above uses a character-level convolutional neural network (CNN) to represent words of a text string into raw word vectors

These raw word vectors act as inputs to the first layer of biLM

The forward pass contains information about a certain word and the context (other words) before that word

The backward pass contains information about the word and the context after it

This pair of information, from the forward and backward pass, forms the intermediate word vectors

These intermediate word vectors are fed into the next layer of biLM

The final representation (ELMo) is the weighted sum of the raw word vectors and the 2 intermediate word vectors

As the input to the biLM is computed from characters rather than words, it captures the inner structure of the word.

Dropout

The dropouts randomly removes the neuron units in the dense layers and asks other neurons in the layer to step up and learn. This is believed to result in multiple independent internal representations being learned by the network. The effect is that the network becomes less sensitive to the specific weights of neurons.

Adam

Instead of stochastic gradient descent in which the learning rates are fixed, Adam optimizer was used where the learning rate adapts to the newest data on a realtive level also called as per-parameter learning rate . It is the combination of AdaGrad and RMSprop.

PROCESS

The image features obtained from the InceptionV3 is fed into Global Average pooling.

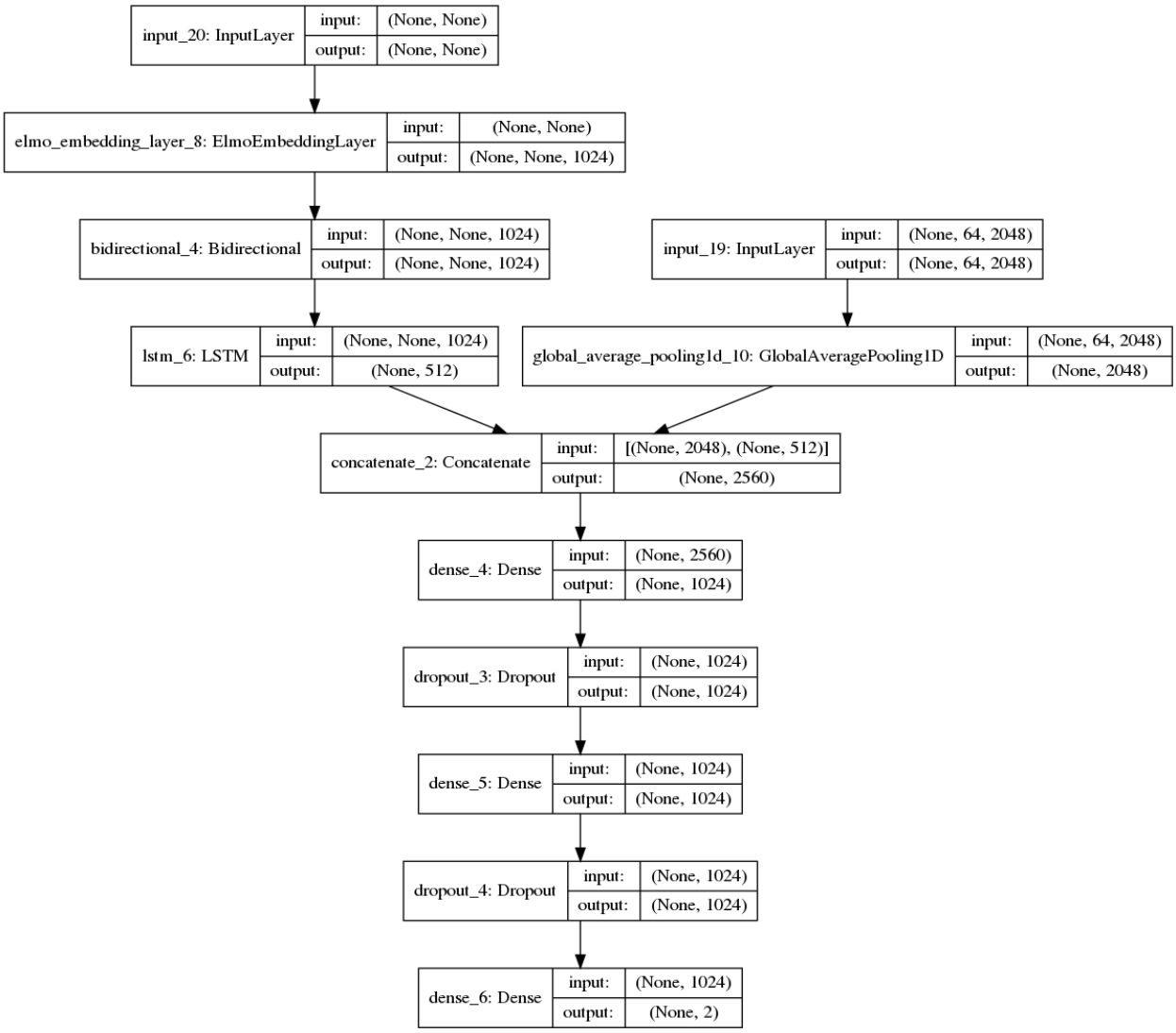
The questions are passed through a ElMo embedding to get the word vectors. The word vectors helps our word tokens to understand the similarity between them. The ones closer in meaning has very high cosine similarity.

The embedding is then passed through a bidirectional LSTM's, the first one returns the sequences, the second one doesn't. The language model is then concatenated on the last axis with the Image result from Global Average pooling. This combines both the image and language model.

To perform prediction on this, we have to construct fully connected Dense layers, with the last being 2 units. The fully connected layer is triggered by softmax activation to assist label prediction.

I used the categorical loss for the loss measure, Adam as the optimizer, trained for 10 epochs over 5 hours.

The accuracy achieved is 60.027%.



Refinement

I used a learning rate of 0.01 initially for the Bidirectional LSTM model, which shot the loss to 9. So I changed it back to 0.001. I dabbled around the number of neurons needed in the dense layers. Initially used 512, got an accuracy of about 54% in 5 epochs. Then used 728, got an accuracy of about 56% in 5 epochs, seeing this increasing trend, I made it to 1024, which yielded 57% in 5 epochs.

I changed the accuracy to Stochastic gradient descent between the training in Double Bidirectional LSTM, it slowed down the training. So I reloaded the saved model from the checkpoint.

Initially, the training loss and validation loss were off a little and started to overfit, so I added dropouts in between the Dense layers to mitigate.

I had a dense layer instead of a global average pool, which added more parameters and also slowed down the training, so I switched it with global averaging pool.

In the attention model, I initially had double attention, one between image and question, another between question and answer. I removed the attention after that, as it was already huge to train.

Initialy, I used my own embedding layer instead of using word_vectors, the training speed was very slow and it didn't seem like anything. It would take days to understand the language constructs ground up. So I switched to GloVe vectors, which performed decently . Later I switched it to the ElMo with great overhead in implementation.

Model	Benchmark model	Un optimized model	Optimized model
Accuracy	50.39%	58.12%	60.027%

Results

Model Evaluation and Validation

During the model development phase, the validation data was used to evaluate the model.

We used 20% of the training set which is 16000 samples of input as the validation set. We didn't use cross-validation as it is expensive in deep learning. The final model architecture and hyperparameters were chosen because they performed the best among the tried combinations. The validation loss was used to save the best model weights.

The validation accuracy for the final ElMo Bidirectional Lstm model is

Justification :

The final model achieved a classification accuracy of 60% on the testing data which is better than what I expected as the benchmark model was just 50 % .

As the word input to the biLM of is computed from characters convolutions of ELMO rather than words, it captures the inner structure of the word. For example, the biLM will be able to figure out that terms like and beautiful are related at some level without even looking at the context they often appear inUnlike traditional word embeddings such as word2vec and GLoVe, the ELMo vector assigned to a token or word is actually a function of the entire sentence containing that word. Therefore, the same word can have different word vectors under different contexts.[3]

The authors of [3] applied ELMO to The Stanford Question Answering Dataset (SQuAD) and claim that After adding ELMo to the baseline model, test set F1 improved by 4.7% from 81.1% to 85.8%, a 24.9% relative error reduction over the baseline, and improving the overall single model state-of-the-art by 1.4%

This structure along with the pretrained weights of the inception model which is constructed by the combinations of the different filter size with residual would in itself form a decent model for the task at hand. But as we have a huge amount of data, adding bidirectional LSTM layers on top of them has helped to avoid overfitting that commonly occurs on the large dataset. The Bidirectional LSTM not only takes the token from the previous time step but also from the rest of the question this makes it understand the context better. Together with ELMO which is called deep contextualized word representation makes both the ideal candidate for the VQA.

In the last few years, experts have turned to global average pooling (GAP) layers to minimize overfitting by reducing the total number of parameters in the model. Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor

GAP layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions h×w×d is reduced in size to have dimensions 1×1×d. GAP layers reduce each h×w feature map to a single number by simply taking the average of all hw values.

Imagine I show my network images of dogs all in the right hand side. So now my final activation map before flatten layers will only have high activations on right side and

the connections to flatten will be trained to only have high weights in that region. So if my dog is now on left, that region does not have high weights to fc layer and is not detected.

By doing a global average we make the convolution invariant to where the object of interest is and this acts as a data augmentaiton of moving the object around to different regions. This prevents overfitting of the fc layers.

The dropouts randomly removes the neuron units in the dense layers and asks other neurons in the layer to step up and learn. This is believed to result in multiple independent internal representations being learned by the network. The effect is that the network becomes less sensitive to the specific weights of neurons.

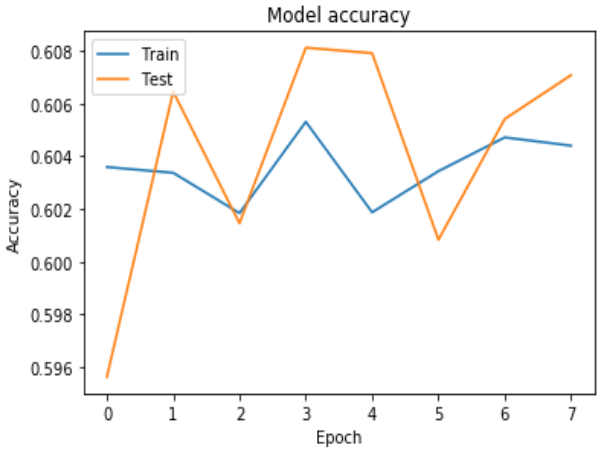
Instead of stochastic gradient descent in which the learning rates are fixed, Adam optimizer was used where the learning rate adapts to the newest data on a realtive level also called as per-parameter learning rate . It is the combination of AdaGrad and RMSprop.

Famous researcher sebastian ruder claims and I quote

“Insofar, RMSprop, Adadelta, and Adam are very similar algorithms that do well in similar circumstances. […] its bias-correction helps Adam slightly outperform RMSprop towards the end of optimization as gradients become sparser. Insofar, Adam might be the best overall choice.”

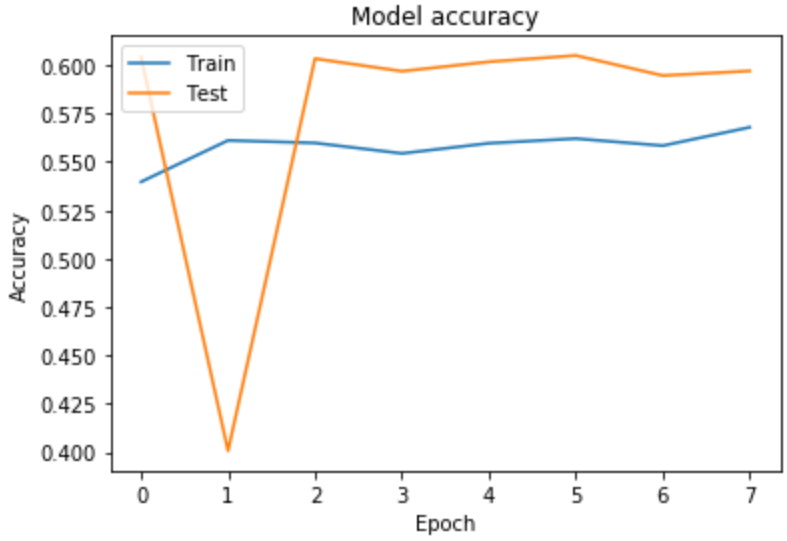
We do not know how the model would perform on Real-word images as the accuracy is just 60% but Visual questioning answering system is not yet used in any professional setting because it is still in the preliminary stage. Combining an image with text will surely allow computers to become more human.

Free Form Visualization:

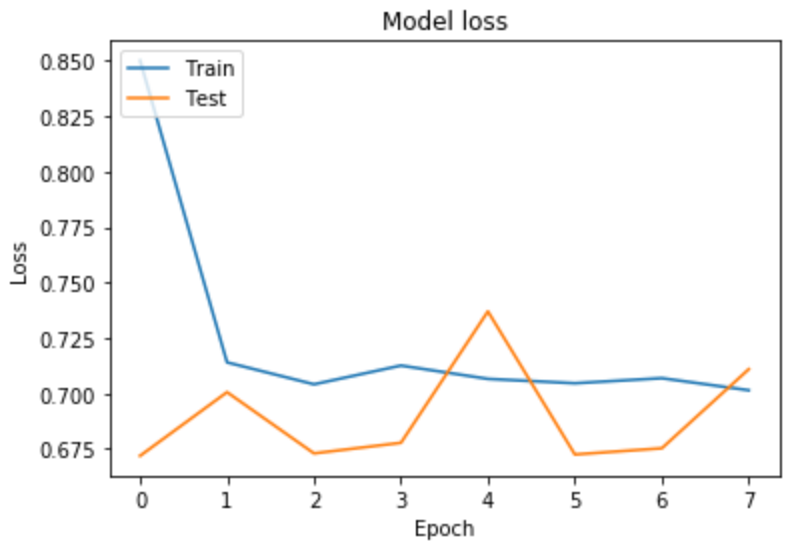


Z

This is the loss and accuracy for epochs 1 to 8



This is the accuracy and loss epochs 8 to 16



This shows that the model had good initial guess of the input but later drowned. Which was pulled back again after tweaks and hyperparameter shifts.

Reflections

I decided to go with the bidirectional LSTM model because the attention model had a lot of parameters to train, and the custom training got out of hand very quickly, the loss rocketed to 87.0 after few epochs. I believed that the attention model would greatly increase the accuracy because of its focused attention to specific parts of the image based on the temporal information of the text. But then I realised it later it is a flawed model and flawed thinking. Attention is known to have improved language models because their encoders are language models, while ours is an image. Such kind of attention model would be used in image captioning where the temporal information is necessary. Ours is a classification task and need not need attention. I wasted weeks on the custom layers,models and custom training of the attention model. Tensorflow 2.0 is very new and had not much community involvement yet, it took a long time to read the documentation to figure things out, so I switched back to tensor flow 1.0 for the final model.

The Interesting aspects of this are the chance to work with a massive dataset even though time consuming and resource killing. The real world problems might have huge amount of data and now I got some foothold on it. One another interesting aspect is how my experiences proves the occams' razor “You don’t need a gun to shoot an ant”. I believed a complex model would be better for the problem. It turns out, the simpler one is better.

The model had problems in prediction due to memory leaks caused by the pretrained model. It was a total havoc which I couldn’t figure solution for 2 days. This along with custom training was the most difficult part of the project.

The final model was way better than the previous models which performed very badly. It cannot be used in any professional setting.

The project went through the following process.

1. The dataset was chosen from vqa dataset.
2. The dataset was explored to find anomolies
3. A sub problem was chosen to solve.
4. The dataset was properly preprocessed and potential data structures were considered.
5. The images were trimmed and the InceptionV3 features were stored.
6. The baseline model was executed
7. The attention model was analysed and researched for few days
8. The attention model was executed with custom layers in tensorflow 2
9. The attention model was droped
10. The LSTM model was considered, a protype was tried and failed due to data wrangling issues
11. The LSTM model was executed but saving and loading process model failed.
12. The word vector performed decently
13. The word vector was replaced with ElMo and ElMo integration with keras was handled
14. The model was tuned with different learning rate and optimizers mechanisms
15. The model had problems in prediction due to memory leaks caused by the pretrained model.
16. The Report was written.

Improvement

I would write the necessary preprocessing tools first hand and employing revision learning. The model can be improved in so many ways. We can add Batch Normalization which would normalize the activations, so that it does not sky rocket . The model can trained for a longer time to eliminate the bias. Deep learning models are full of hyper-parameters and finding the best configuration for these parameters in such a high dimensional space is not a trivial challenge. We can use random search to pick the best values of weight decay, units of neuron and Kernel intializers. We can also tune the momentum of the Adam optimizer. But that would all take days to tune. The learning rate can be tuned by writing custom schedulers. There is room for plenty of improvements. Given the time constraint I tuned them to the best of my ability.

References

[1] Bingbing Liu, Weini Yu -Scaled Attention for Visual Answering https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6909201.pdf.

[2] Aishwarya Agrawal* , Jiasen Lu* , Stanislaw Antol* , Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh - VQA: Visual Question Answering https://arxiv.org/pdf/1505.00468.pdf

[3] Matthew E. Peters†, Mark Neumann†, Mohit Iyyer†, Matt Gardner†, -Deep contextualized word representations

[4] Peng Zhang*† Yash Goyal*† Douglas Summers-Stay‡ Dhruv Batra† Devi Parikh††Virginia Tech ‡Army Research Laboratory - Yin and Yang: Balancing and Answering Binary Visual Questions

]https://arxiv.org/pdf/1511.05099.pdf