Introduction - James

The task our model performs is image captioning. So it takes an image as input, and outputs a caption for that image. The captions can vary in length, so not all captions generated by our model will have the same number of words. The caption aims to describe details about objects and actions present in the image. For example, if an image contains a person eating pizza at a table, the caption should make mention of that.

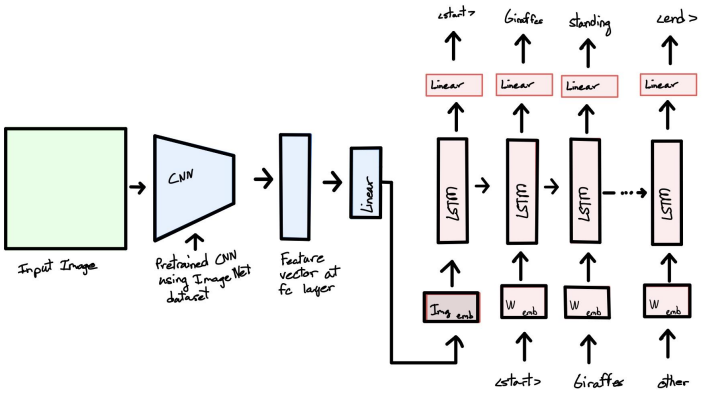
The deep learning model we used to perform this task is an LSTM (recurrent neural network) combined with a CNN to generate captions for images. The CNN encodes the image into a word embedding, which is passed to the LSTM as input for the very first timestep. The LSTM outputs a sequence of words, one word for each timestep, until it reaches the maximum caption length, or it produces a token that indicates the end of a caption.

During training, the LSTM uses teacher-forcing, meaning that even if it outputs an incorrect word for a given timestep, it will still use the correct word as input for the next timestep.

We do not use teacher forcing when checking test set accuracy and generating captions for images from the test set, meaning we always use the word generated from the previous timestep as input for the next timestep.

Model

Model Figure - Mautasim



This figure is a rough sketch of how the model architecture is designed for this deep learning model. The model first takes in an image and passes it through a pretrained alexnet CNN model. The last layer of the pretrained CNN model, which is a softmax layer, is removed as we need to connect the CNN model to an LSTM model to properly generate a sequence of captions.

Since the last layer of the CNN model is removed, we are left with a feature vector which we pass into a fully connected linear layer, which transforms the feature vector into a word embedding. The word embedding is then passed into the LSTM Model which outputs a word in for every timestep. The first timestep output is the <start> token and the caption ends when a timestep outputs the <end> token. Furthermore, to get the actual word, we get the argmax of the

logit vector which we get after applying the linear layer after the LSTM. For each timestep, the LSTM uses a word embedding of some word as input.

Model Parameters - Mautasim

```
+---------------------------+------------+
|          Modules          | Parameters |
+---------------------------+------------+
|   encoder.linear.weight   |  2304000   |
|    encoder.linear.bias    |    250     |
|    decoder.embed.weight   |  2229500   |
| decoder.lstm.weight_ih_l0 |   300000   |
| decoder.lstm.weight_hh_l0 |   360000   |
|  decoder.lstm.bias_ih_l0  |    1200    |
|  decoder.lstm.bias_hh_l0  |    1200    |
|   decoder.linear.weight   |  2675400   |
|    decoder.linear.bias    |    8918    |
+---------------------------+------------+
Total Trainable Params: 7880468
7880468
```

The table above is a summary of the parameters associated with the model. Since we are using transfer learning, we do not train the parameters associated with the CNN model except for the last linear layer (since that was something we added). The column "modules" defines where the parameters come from the model and the column "Parameters" defines how many trainable parameters are associated with that respective layer. The total number of trainable parameters for this model is: 7880468.

Model Examples - Arjun

The following is an unsuccessful prediction, with a BLEU score of 0.11.
True Caption: a black and white dog is running through a cow field .
Generated Caption: a black dog is jumping over a pool .

Unsuccessful prediction

The following is a successful prediction, with a BLEU score of 0.59
True Caption: a black dog is running in snow.
Generated Caption: a black dog is running through the snow.

Successful prediction

Data

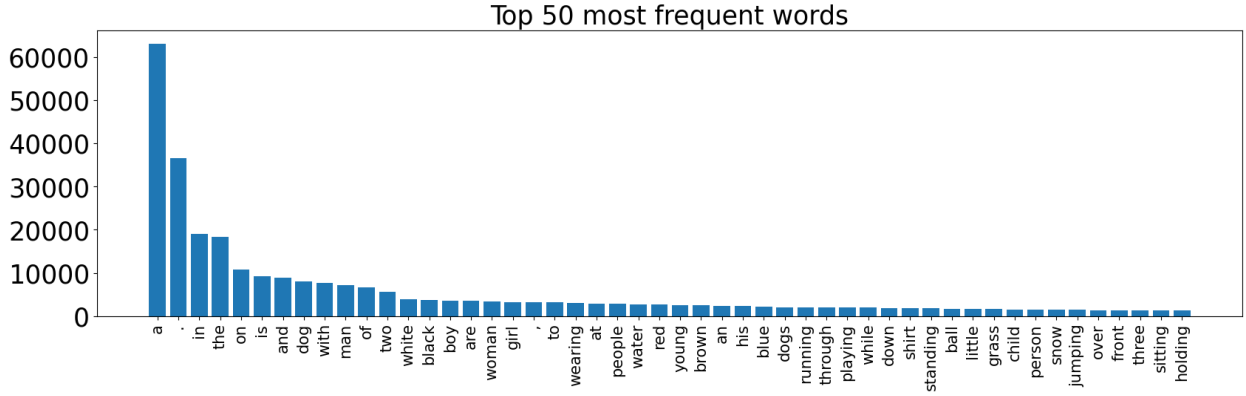
Data Source - James

The images are from the flickr8k dataset, which contains 8000 images, and each image comes with 5 captions that describe the image. The 5 captions are worded differently, but basically mean the same thing. The 8000 images are not all of the same action/object, it covers a wide variety of objects and actions that are often found in the types of photos people usually take. This can be seen in the data exploration section of our code file.

We chose this dataset because it has a smaller size than a lot of the other image captioning datasets. This smaller dataset size allowed us to train more efficiently when searching for the optimal hyperparameters. Additionally, this is a common dataset used for image captioning, meaning we have other models trained and tested on this dataset to compare with.

Data Summary - Scott

For the data summary, we provided a data exploratory section in our code file, using the captions that correspond to an image. We created a matplotlib word frequency bar plot showing the top 50 most common words across all of the dataset's captions, to help us understand the properties of the dataset.



This tells us the model should be using these words frequently, since they are very common in the dataset, and using them frequently would be necessary for high accuracy. This also means our model should be good at being able to tell what a "dog" is and what a "man" is (among others) since the chart shows us there are many images and captions involving these.

We also found the percentage of the time each word appears (for some of the most common words)

```
a 13.21338518919418%
. 7.673702449727925%
in 3.980440774817183%
the 3.863807042495794%
on 2.2538000360809387%
is 1.9603277491787392%
and 1.8569097095484428%
dog 1.706712313249676%
with 1.6288865674021304%
man 1.5242098903726826%
of 1.4082054767508696%
two 1.182909382302719%
white 0.8265052254429355%
black 0.8038497522582053%
boy 0.7511967543936934%
are 0.7352540140044388%
woman 0.7138571782188603%
```

This means we can expect our model to produce ‘a’ about 13% of the time, ‘.’ 7% of the time, and so on.

From our exploratory data we also found that the average description is about 11 words long. This means we can expect our model to finish generating its caption in about 11 words on average, since that’s what it will be trained to do.

Additionally, our vocabulary size is about 8900. The entire English language vocabulary size is much bigger than this, so this means we can’t expect our model to perfectly describe every possible image that can be captioned in English. The words it generates are limited to this 8900 words long vocabulary, so our model can only describe images that can be accurately captioned using a subset of these 8900 words.

Data Transformation - Scott

Each image came with 5 captions, and we decided we would use only 1 out of these 5 for the target caption.

We found that the longest caption has a length of 40. Using the maximum caption length, we added the necessary whitespace padding tokens (<pad>) to each caption in the data set such that each caption’s length is equal to the maximum caption length. We also added <start> and <end> tokens to the captions to indicate the start and end.

We converted the words in captions to unique identifiers (indices) so that our neural network (LSTM) is able to generate captions using word embeddings based on the indices of the words.

Images were transformed to normalized 3x224x224 tensors and captions (list of indices) were transformed to tensors.

We additionally augmented the data (added new data points) by creating a copy of each image, but slightly darker. This is because it appears as though many of the images from the dataset are taken during the day, and we want our model to perform well for pictures taken closer to night time too. It should also be able to give the same description for an image, even if it's slightly darker. So for each image, we made a new data point, which uses a slightly darker version of the image with the same caption. So this made the dataset become 16k images instead of 8k.

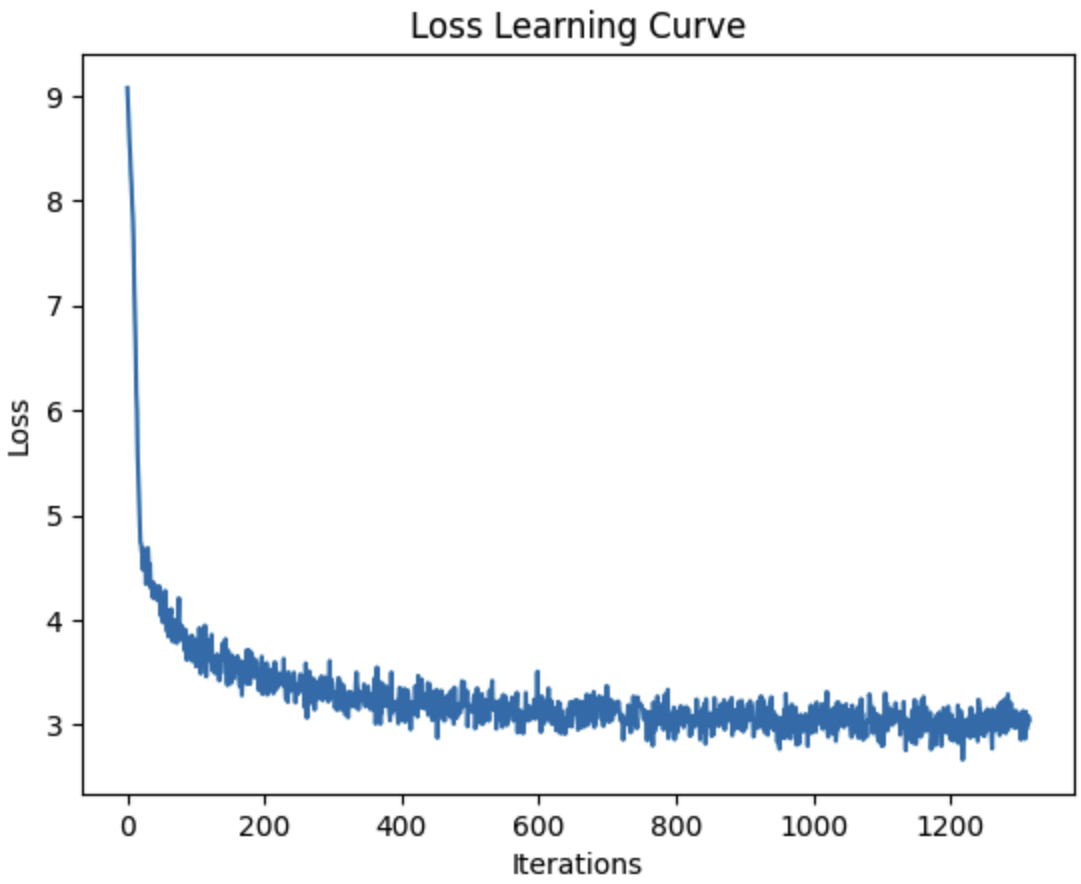
Data Split - Scott

The data set to split the train/validation/test data apart was predefined randomly. The flickr8k dataset consists of 8000 images. We used in total 8091 images to split between the training set with 6000 images , the validation set with 1000 images, and the test set with 1091 images. With the data augmentation, each of these numbers doubled. Overall, the training, validation, and test sets were defined randomly. The randomness helps reduce the chances of certain patterns being present in only certain sets. We want the objects/actions/patterns present in these images to be spread out evenly across the datasets, so that each dataset can be more representative of the others. This helps our model's ability to generalize past the training set.

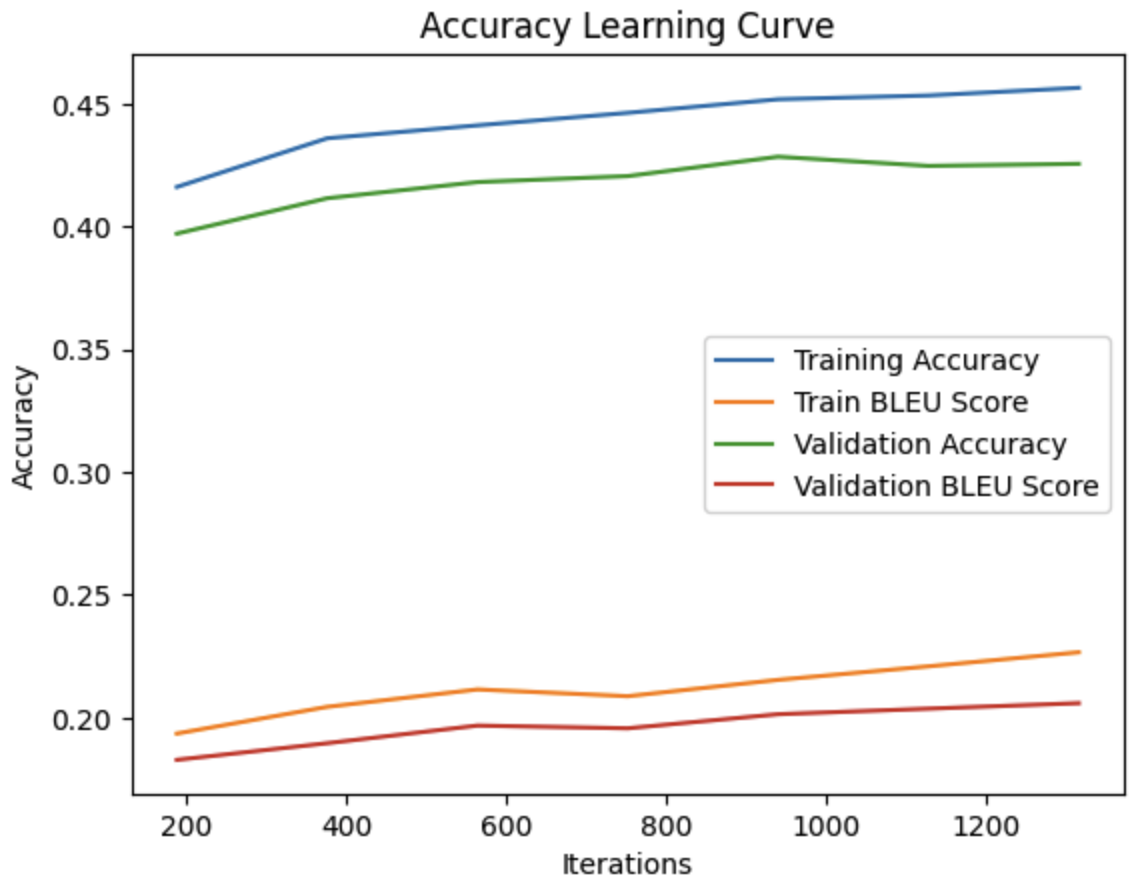
<u>Training</u>

Training Curve - Arjun

The following curve shows how our loss changed over time. Although some iterations slightly increased the loss of our model, the data trends downwards, indicating a successful and reasonable model.



This graph illustrates the increase of accuracy, both in the training and validation data, through increased iterations. We measured the accuracy of our generated captions by calculating the percentage of correct words in our generated caption vs. the true captions. We also measured the BLEU score of both our training and validation set, which strongly correlates with the other accuracy measure, as depicted below.



Hyperparameter Tuning - Arjun

To tune the hyperparameters of our model, we used the grid search algorithm to thoroughly test a large combination of hyperparameters. First, we define a dictionary of each hyperparameter value we want to test, and manually input each possible value that we want the algorithm to consider. The advantage of grid search is that although running the algorithm is computationally expensive, it exhaustively tries each possible combination of hyperparameters. The procedure is extremely thorough. The following table shows the hyperparameters that were tested, and the values tested for each:

Hyperparamter	Values
batch_size	16, 32, 64, 128
weight_decay	0.0, 0.1, 0.01, 0.001
learning_rate	0.1, 0.01, 0.001, 0.0001, 0.00001
num_epochs	3, 5, 7, 9

We determined the best combination of hyperparameter values by both looking at our validation accuracy, which is calculated by finding the number of correct words in our description, and the average BLEU score of the model. Unsurprisingly, the two values were strongly correlated. The highest validation accuracy and BLEU score was found with the following hyperparameters:

- batch_size = 64; this makes sense because a smaller batch size might be too noisy, while the larger batch size might not have generalized well.
- weight_decay = 0.001; our model performed the best with this weight decay, as the regularization helps the model avoid overfitting and learn nuanced details in the image dataset
- learning_rate = 0.001; model seemed to have converged the best with this smaller learning rate without overshooting an optimal solution
- num_epochs = 7; this seems to be the best number of epochs to train on without any possible under or overfitting

Results

Quantitative Measures - James

The measure we chose for the training loss was cross entropy loss. This is because we want the correct word, out of all words in the vocabulary, for each timestep.

For training and validation accuracy, we looked at the percentage of words in correct spots during training (so with teacher-forcing). This helped us get an idea about how good the model was getting at the specific, exact thing it was being trained to do: minimize its loss, which mainly happens through predicting as many correct words in correct spots as possible when being teacher-forced. We also kept in mind that being good with teacher forcing would likely correlate to being decent without it.

But we know there's more to the meaning of a sentence than just the number of correct words in correct spots, so we also looked at the bleu score. The bleu score can measure the similarity between the output caption and the target caption, and it's better at accounting for how there are multiple correct ways of describing things, even when we don't have every word in the correct spot.

We chose to look at both the bleu score and the percentage of words in correct spots because the percentage of words showed us how good the model was getting at its specific task, and the bleu score showed us how similar the captions are. Both of these pieces of information are informative, and we want to optimize both of these.

For test set performance, we looked at the bleu score and percentage of correct words in correct spots, but without teacher forcing since we don't want the model to use target captions at all when tried on images in the real world.

So in short, for training loss, we chose cross entropy, for validation performance, we looked at percentage of correct words and bleu score with teacher forcing, and for test set performance, we looked at percentage of correct words and bleu score without teacher forcing.

Quantitative and Qualitative Results - James
After training, our model was found to have the following:

Training loss	3.04
Average training accuracy	46%
Average training bleu score	0.23
Average validation accuracy	43%
Average validation bleu score	0.21
Average test accuracy	33%
Average test bleu score	0.08

The training curve was noisy due to how we used relatively small batches of data for iterations of gradient descent.

Overall, the model was able to make reasonable captions when tested on examples from the test set. It often got the main objects and colors of the images correct. It sometimes got the main action correct as well, but not always. Our model was also able to place the <start> token at the correct spot every time, and also placed the <end> token reasonably close to the right spot. The model had a tendency to repeat itself, particularly with variants of captions from the training set.

Justification of Results - James
These results make sense given the difficulty of the problem. The difficulty comes from needing to generate captions, which is a subjective process, by analyzing an image and producing a

sentence. Each of these tasks is nontrivial on their own for a computer, so combining them can make for a challenge.

Despite the challenging nature of captioning images properly, our model was still able to produce reasonably good captions for images from the test set, as described above. However, our model and the problem it's trying to solve have some issues which need to be discussed. These issues help explain the qualitative issues our model experienced when generating captions for images. They additionally explain our model's suboptimal accuracies and bleu scores.

Captioning images is highly subjective, since one person's idea of an accurate caption for a given image may differ wildly from another's. There are so many different ways to describe an image, so when a model is trained to only caption images in one particular way with one exact caption to match, it's being trained to only have one viewpoint. This can limit the expressiveness and flexibility of the model because it's being trained to think that alternative captions are wrong, even if they're accurate. Limiting the model in this way can worsen overall performance.

Additionally, our quantitative measures for measuring sentence similarity are good, but limited since they won't be as good as human judgment. There's only so much we can do to measure the similarity between two sentences by getting the computer to do it. It's not practical to manually look at each caption when measuring similarity, so we had to go with the bleu scores and accuracies the computer gave us when deciding on an optimal model. It's possible the computer gave model A worse accuracies and scores than model B, but if model A had better captions (based on human judgment), that is something we would have missed. This means it's possible we chose a suboptimal model when tuning hyperparameters.

On the test set, the model couldn't rely on teacher forcing, and it appears that not being nudged in the right direction caused it to sometimes get thrown off early and not really know how to recover. After all, the model was trained to minimize loss with teacher forcing. For example, let's say the correct caption is "a man stands in a hallway", and the model generates "a man sits", with more to generate, then it's not learning how to recover from that. This is because without teacher forcing, the word "sits'" would be used as input for the next timestep during training instead of "stands". It would only be trained to produce "in" when receiving "stands", and not "sits." So the word "sits" could really throw it off. This would explain why the accuracy with teacher forcing was better than without.

Additionally, the model could be memorizing training captions rather than actually learning how to generate them. This would explain the repetitive behaviour on the test set, where it sometimes seems to just spit out the start of a training caption that gave it good results during training, and just goes from there. In particular, "dog" and "man" often occurred at the start of these repetitive captions, which is not surprising because these are among the most common words in the dataset

according to our chart about the most frequently occurring words. So of course predicting these words often would allow for good training accuracy, which explains why we see these words a lot when the model is tested. This is basically overfitting, since the model is memorizing specific words and phrases from the training set that allow it to do well during training, but the model can't generalize super well. This overfitting may be the result of hyperparameter choices, such as the small weight decays, among other factors. The lack of teacher forcing plays a role here as well, since performing well during training may only require memorization of captions with the help of teacher forcing. But when performing on unseen data, we can't rely on memorizations and teacher forcing to produce a good result.

Ethical Considerations - Mautasim

A system that can generate useful captions given an image is a tool that is being used widely in today's era. However, it is important to mention the possible use cases that can cause severe ethical issues if not used properly and should be used with a high level of responsibility. Possible use cases of this model that can cause ethical issues are in the law enforcement and surveillance field, social media platforms and in health care settings.

Surveillance cameras are widely used today to ensure security and the safety of an area. If a deep learning model can generate captions containing sensitive information on identity, religion, race or political views on any given person without their consent, it could potentially breach someone's privacy.

Another possible ethical issue that can arise is when the model is used and the information revealed is spread through social media or a news outlet as the captions generated can be incorrect, which can spread misinformation and hate speech. This can bring back negative biases and stereotypes towards minority groups, extending to discrimination and social inequality. This can happen if the training set is biased and only contains images and/or captions making certain minority groups look bad and other groups of people look good.

Lastly, the use of this model in a health care environment can also raise ethical issues revolving around protection of a patient's data and privacy. If the model is used to generate information based on images of a medically admitted patient, sensitive information can be leaked regarding the patient's medical history and the patient's health issues which should remain confidential.

Authors

James: Wrote functions for training the model and for obtaining the percentage of words in correct spots. Got the model to overfit on a small dataset. Additionally wrote a function for getting the model to generate a caption without teacher forcing. Helped with transforming the data into a usable form for the model by creating the dataset class, tensor transforms, and data loaders. Also augmented the data. Made some small fixes to model architecture to get it to work. For the readme, I wrote for intro, data source, quantitative/qualitative results, and justification of results.

Mautasim: Designed and implemented the required model architecture for the task. Created classes “EncoderConvNet", “DecoderRecurNet” and “EncoderToDecoder” to successfully implement the corresponding parts of the model. I was responsible for including the Ethical considerations, Model figure and Model parameters in the report.

Scott: Created functions that converted the captions for each image to its necessary form (words to indices), added the necessary starting, ending, whitespace padding to each caption in the dataset for the neural network to be trained on. Created a dataframe to show the relation between an image file with its caption and translated caption to indices. Added any necessary changes to the data frame that may consist of invalid image filenames. For each data frame row, I’ve converted them to a list of pairs, where the pairs consist of the image file path and a list of padded captions to indices. These pairs were used so that the image file and caption list Created a visual representation of what captions correspond to a specific image. Assisted with writeup questions: data summary, data transformation, and data split.

Arjun: Incorporated BLEU metrics to our accuracy predictions, and modified our training graphs to visually depict the BLEU score increasing over time. Created a new function that gets the accuracy of a dataset for a given model, but without the use of teacher forcing. Tuned the hyperparameters and created the code that used the graph search algorithm to find the perfect set of parameters. Generated test accuracy and outputted correct and incorrect prediction examples. For the writeup I was responsible for sections: hyperparameter tuning, training curve, and model examples.