LIMITATIONS

- Most machine learning models learn a specific task. For example, an image classifier trained on classifying dogs and cats is expected to do well only on the task that we have given, which is the task of classifying dogs and cats. We generally would not expect such a machine learning model to be very good at detecting any other animals such as raccoons. So we can successfully say that standard vision models are generally good at doing one task and one task only.
- The datasets that are used for computer vision are generally labour intensive and costly to create. These computer vision datasets require significant additional human time for labelling images.
- So now we know that most state-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. Putting it again in layman terms, an image classifier trained on classifying dogs and cats is expected to do well only in classifying dogs and cats, and not in detecting any other animal like raccoon. This restricted form of supervision limits their generality and usability and makes them inflexible in general; since additional labelled data is needed to specify any other new class and also, additional steps are required for fine-tuning the model or training a new classification layer.

INTRODUCTION TO CLIP

- CLIP is a shorthand representation for Contrastive Language-Image Pre-training.
- It is a neural network model trained on a wide variety of images and captions that are abundantly available on the internet. Since CLIP is built on images and captions that were already available on the internet, it significantly cuts off human time spent on labelling those dataset images.
- Standard classification models completely discard the semantic meaning of the class labels and simply enumerate numeric classes behind the scenes; whereas CLIP works by understanding the meaning of the classes. CLIP expands knowledge of classification models to a wider array of things by leveraging semantic information in text.
- Learning directly from raw text about images is proving to be much more promising than the already available standard supervised models. CLIP has impressive **zero-shot** capabilities, making it able to accurately predict entire classes that it has never seen before! In our CLIP model, natural language is used to reference learned visual concepts or to describe new visual concepts. This enables zero-shot transfer of the model.

CONTRASTIVE LEARNING

Many of us may remember solving this kind of puzzles where it is asked to match the image-text pairs on the two sides by drawing lines like this. In a similar fashion, CLIP has learned to basically do this job by matching a much larger number of image-text pairs using a huge dataset.

PREDICTIVE APPROACH

Now let us get a brief idea as to what is this predictive approach. Suppose we have a bag full of words, like the one shown in the image. Let’s say, some of the words in the bag are: are, cat, dog, is, now, on, table, the and so on. Now, we basically need to pick up some of the words from the large vocabulary available here. We need to choose those words that can successfully describe the picture given in the slide. So, for describing this image, the model now picks up some words such as dog, is, on, table and the. Now, if we reorder the words, we get a perfect caption out of these, which can successfully describe the image above. So, basically we get a caption for the image like “the dog is on the table”.

CONTRASTIVE APPROACH

On the other hand, in contrastive approach, we have an image of let’s say, Siberian Husky, and corresponding to

this image, we have multiple text captions to describe this image. The captions read “a photo of a siberian husky”, “a photo of a german shepherd dog”, “a photo of a collie”, “a photo of a border collie”, “a photo of a rottweiler” and so on. Now, among these captions available, our model needs to figure out which text caption best represents the given image on the left. It does this by assigning certain values of probabilities to each caption. The blue bars on the right shows these probabilities and we can easily observe that the highest probability has been assigned to the first text caption, which reads “a photo of a siberian husky”. In this way, our model has correctly chosen the most appropriate label for the image given on the left.

So, we can conclude that contrastive approach is much more data-efficient than using predictive approach and this is the reason why a contrastive approach was chosen over a predictive one. Given an image, predictive methods have to learn to predict the exact words that can describe that image. This often becomes a difficult task due to the wide variety of descriptions, comments, and related text that co-occur with images. Initially, the researchers had given a try at predictive approach but they found that there were difficulties in efficiently scaling this method. In contrast to this, contrastive methods just need to determine if the image-text pairs are matching or not. Recent works in contrastive learning for images have found that

contrastive objectives can learn better representations than their equivalent predictive objective. As a result of this analysis, contrastive approach was chosen for CLIP.

CONTRASTIVE PRE-TRAINING

- We start with the inputs consisting of paired text and images. At each iteration, a certain batch size of these pairs is randomly sampled from a large dataset.
- The model is composed of two encoders, one for each modality. The text encoder encodes its text examples into text feature vectors as we can see here. The text cards shown on the left of the image are put into the text encoder and then these text descriptions get converted into their corresponding text feature vectors T1, T2, T3,... upto TN.
- And similarly, the image encoder outputs image feature vectors corresponding to the input image. The image cards shown on the left are put into the image encoder and then these images get converted into their corresponding image feature vectors I1, I2, I3,... upto IN.
- Moving next, we use the text feature vectors T1, T2, T3… and image feature vectors I1, I2, I3… to make a grid like this. Each cell of this grid represents the similarity between a particular image-text pair. For example, here we are having I1.T1, I1.T2, I1.T3,... upto I1.TN. Then we are having I2.T1, I2.T2, I2.T3

and so on. Like this, we can calculate the cosine similarities between each pair of vectors where the similarity values for the matching pairs of text and image inputs are highlighted in blue. As we can see, the diagonal elements of grid, like I1.T1, I2.T2, I3.T3,... upto IN.TN are highlighted in blue since these cells represent the matching image-text pairs and hence, similarity values are highest in these. For contrastive pre-training, we want these highlighted similarity values of the diagonal cells to be higher, indicating the matching text and images are mapped to a similar region in the feature space. On the same note, we also want to keep the similarities in the rest of the cells as low as possible, since these text and images do not match.

- For instance, if we look at the second image feature (I2) and the corresponding cosine similarities, we have a classification problem as to … which text description fits the best corresponding to this image, where the second text feature (T2) is the correct answer, just because image vector I2 matches the most with the text feature vector T2.
- Also here, if we look at the third text feature (T3) and the corresponding cosine similarities, we have another classification problem as in … which image is the best match for the text description that we are talking about, where the third image feature (I3) is the correct answer, again because text vector T3 matches the most with the image feature vector I3.

EMBEDDING

Suppose we have one cat and two dogs. We can represent this data as a dot on a graph, like this. Similarly, we can think of embedding as a way to smash information into mathematical space. We just took information about dogs and cats and smashed it into mathematical space here. Similarly, ***We can do the same thing with text and with images!***

For this purpose, the CLIP model consists of two sub-models called encoders:

- a text encoder that will embed or (smash) text into mathematical space, and
- an image encoder that will embed or (smash) images into mathematical space.

GOODNESS & BADNESS

Whenever we fit a supervised learning model, we have to find some way to measure the "goodness" or the "badness" of that model – the goal is to fit a model that is as "most good" and "least bad" as possible.

The CLIP model is no different: the text encoder and image encoder are fit to maximize goodness and minimize badness.

So, how do we measure "goodness" and "badness?"

In the image shown here, we can see a set of purple text cards going into the text encoder. The output for each card would be a series of numbers.

For example, the top card, pepper the aussie pup would enter the text encoder. This text encoder will now smash the text description into mathematical space, and the output will come out as a series of numbers. For instance, let’s take it as (0, 0.2, 0.8).

The exact same thing will happen for the images also. Each image will go into the image encoder and the output for each image will also be a series of numbers.

The first image card, with Pepper the Aussie pup drawn on it, would enter the image encoder. This image encoder will now smash the image into mathematical space, and the output will come out like another series of numbers like (0.05, 0.25, 0.7).

If we look closely, we can observe that the numbers coming out of the two encoders are very close. One is (0, 0.2 and 0.8), the other is (0.05, 0.25 and 0.7). This shows that the text description for pepper the aussie pup matches with that of the image.

COSINE SIMILARITY

In an ideal world, the series of numbers for the text

"pepper the aussie pup" will be very close (identical) to the series of numbers for the corresponding image. *In fact, this should be the case everywhere*: the series of numbers for the text should be very close to the series of numbers for the corresponding image. One way for us to measure "goodness" of our model is how close the series of numbers for each text is to the series of numbers for each image.

There is a convenient way to calculate the similarity between two series of numbers: the cosine similarity.

The cosine similarity is basically cosine of the angle theta between the two vectors A and B, as shown in the image. This can be mathematically represented as the dot product of vector A and B, divided by the product of their magnitudes.

GOODNESS & BADNESS

In the image shown, the blue squares on the diagonal represent where the text and image coincide. For example, T1 is the embedded representation of the first text; I1 is the embedded representation of the first image. We want the cosine similarity for I1 and T1 to be as high as possible. We want the same for I2 and T2, and so on for all of the blue squares. **The higher these cosine similarities are, the more "goodness" our model has!**

At the same time, we want to minimize the cosine similarities for the non-diagonal grey squares. As we can see here, there are a lot of grey squares that indicate where the text and image do not match with each other. For example, T1 is the text "pepper the aussie pup" but perhaps the image I2 represents the image of some other animal such as raccoon.

We want the cosine similarity between this image (I2) and the text “pepper the aussie pup” to be pretty small, because these two do not match!

While we wanted all the blue squares to have high cosine similarities, we want all of the grey squares to have low cosine similarities, because that measures "badness" of our model.

ZERO-SHOT PREDICTION

- As a result of this contrastive pre-training, now we have obtained a set of encoders that can encode text and images into a shared multi-modal embedding space. A powerful application of the multi-modal encoders and embedding space is zero-shot image classification or zero-shot prediction. Suppose we have a set of labels as shown in the image: plane, car, dog, bird etc.
- From these textual labels, we first construct the text prompts or text descriptions. For this purpose, we use certain templates like the one shown here in

this image. The template reads “a photo of a {object}”. Now, we need to replce the object by the respective labels like plane, car, dog, bird. In this way, we get text descriptions such as “a photo of a plane”, “a photo of a car”, “a photo of a dog”, “a photo of a bird” and so on. Constructing text descriptions like this, tends to work better for our text encoder than using just words.

- Now, we feed these prompts to the text encoder in order to obtain text features T1, T2, T3 etc. corresponding to each of the text labels.
- Now, we take an image that we want to classify. For this, first we need to obtain the corresponding image features by using our image encoder. We name this as I1.
- Now that we have the feature vectors for all the labels and also for the input image, we can calculate the cosine similarities between the image feature vector and each of the text feature vectors. After that, we can find the largest similarity value among them.
- This will select the text feature vector, which is the closest in angular distance, to the image feature vector. In our example, the highest similarity value is of the third label, a **dog**. So, basically CLIP has successfully detected that the input image represents “a photo of a dog”. CLIP can perform this classification based just on textual label information without any additional steps for

fine-tuning the model or training a new classification layer. So we call this method zero-shot image classification.

ZERO-SHOT LEARNING

Zero-shot learning is when a model attempts to predict a class it saw zero times in the training data. So, using a model trained on exclusively cats and dogs to then detect raccoons. A model like CLIP, because of how it uses the text information in the (image, text) pairs, tends to do really well with zero-shot learning. Even if the image we are looking at is really different from the training images, the CLIP model will likely be able to give a good guess for the caption for that image.

CLIP ACTS AS BRIDGE

It is said that CLIP is a bridge between computer vision and natural language processing. So let us now briefly understand what is this computer vision and natural language processing.

Computer Vision is "the ability for a computer to see and understand what it sees in a manner very similar to that of humans."

Natural Language Processing is "the ability for a computer to understand language in a manner very similar to that of humans."

Now, the image module in CLIP relates to Computer Vision, and picking up raw text from captions available on internet, relates to Natural Language Processing.

So we can safely say that CLIP definitely acts as a very powerful bridge between computer vision and natural language processing, with a lot of flexibility and a varied range of applications.

RESULTS

Now, let us look at some predictions made by the CLIP model on examples from various datasets.

The first picture on the top left corner shows a building. The text labels corresponding to this image read “a photo of a building”, “a photo of a carriage”, “a photo of a statue”, “a photo of a bag” and “a photo of a mug”. Here, CLIP assigns a really high probability of 97.7% to the correct label, i.e., “a photo of a building”, as opposed to other labels like “a photo of a carriage, statue, bag or mug”.

In the picture towards the bottom left corner, it correctly identifies that the picture corresponds to “a photo of a marimba”, among 1000 labels. Whenever there is a bright blue bar at the top, CLIP has guessed this correctly. And if there is a pale blue bar at the top (like in the bottom right image), it indicates an incorrect label.

With this, let me wrap up for today. Thank you so much!!