Im2Vec : A Language Model Approach to Understanding Image Classification

Irene Lee
Georgia Institute of Technology
ilee300@gatech.edu

Seung Je Jung
Georgia Institute of Technology
sjung323@gatech.edu

Mridul Bansal
Georgia Institute of Technology
mbansal31@gatech.edu

Abstract

In this work we present Im2Vec, a novel approach to understanding the images in image classification task. Whereas conventional methods for weight visualization have tried to generate images from the weights of trained models, we sought to derive a numerical representation– image embedding vectors. The idea is adopted from Word2Vec, a method to encode relationship among words into word embedding vector. Im2Vec uses the pixel values of each image as the context for its classification, similar to the context words in skip-gram model. By doing so, the generated image embedding vectors can reflect the relative importance of each pixel for each class as well as show relationship between different classes via the proximity between vectors of corresponding classes. The code is publicly available at `https://github.com/irenelee5645/image_vector`

1. Introduction

For both language models and image classification models there has been much effort to understand the words and images for each respective task. For language models, methods such as Word2Vec and Glove, which generate word embeddings, are common approach to understand and encode words. On the other hand, for image classification, often times the weights are visually represented to show how different regions of an image relate to a particular class. As such, there isn't much similarity between the approaches in the two fields. In this paper, we have designed Im2Vec, a novel way of visualizing and understanding images in image classification tasks by applying language model approach to images. Specifically, we developed a model to generate image embedding vectors that show relationships among different images, instead of the relationship between an image and its corresponding class as in conventional visualization methods for image classification models. Ideally, objects that are visually similar (e.g. lions/tigers) would have similar embedding vectors, whereas those that are visually distinct would output dissimilar ones. Our work presents a fresh view towards interpreting images in image classification task. By exploring relationship among different images rather than an image and its corresponding class, we hope to reach a deeper understanding of image classification techniques.

2. Related Works

2.1. Word2Vec

The first problem that needs to be addressed when designing any NLP model, or more generally any model that uses textual input, is how words in the input data will be represented. Since words as ASCII text do not have any intrinsic meaning, finding good word embeddings that encode the meanings of words whilst maintaining a format that models can easily use is an important field of study. Word2Vec is one method used today to produce word embeddings that manages to efficiently capture the underlying concepts of words [3].

In Word2Vec, words are mapped into a vector space such that words which are more similar to each-other are closer to each-other in the sense that they have a higher cosine similarity [4]. Therefore, we would expect to be able to see vectors corresponding to similar concepts to appear in distinct groups. The way that this word embedding is produced is surprisingly simple, and can be done in two separate ways: continuous bag of words and skip gram. In continuous bag of words, a 2 layer neural network is used which, given a window of words with the middle word missing, tries to guess what word belongs in the middle [2]. After training on a multitude of textual data, the model will produce a probability distribution of the likelihood that each word in the vocabulary would occur in the middle. After training,

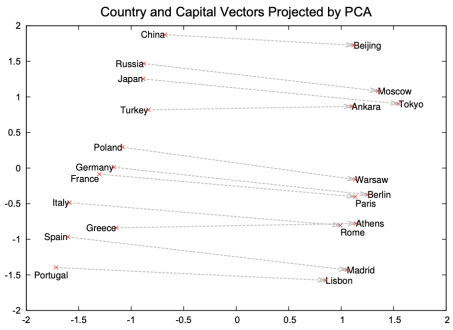


Figure 1: 2 Dimensional PCA of word vectors corresponding to different countries and their capitals. [4]

we can remove the softmax layer, and the word embeddings are the output of the first layer when a word is presented to the model as input [2].

For visualising Word2Vec, PCA can be applied on the output vectors of words to reduce them to 2 dimensions, allowing for the examination of a specific subset of words to check for any relations [4]. For example, in Figure 1, we can see a that words for capital cities and countries are distinctly on separate half-planes, and that corresponding pairs of capital cities and countries are roughly vertically in line with each-other.

2.2. Visualizing Feature Maps and Filters of Image Classification Models

There are many available methods with which the feature maps and filters of image classification models, namely CNNs, can be visualized for insight into what the model is doing. A couple of the most common ways to achieve this is through generating saliency maps using gradient backpropagation, and CAM based methods such as grad cam[1]. Saliency maps are images that represent how important each part of the image is, allowing visualisation of what a model is identifying in an image. This can provide insights into how a model may be producing erroneous results, or improve people's trust in how machine learning models classify images [5]. One prominent way of generating these saliency maps is to use gradient backpropagation, in which the gradients of a network are backpropagated to the original image to identify which parts of the image need to be changed the least to create the most change in the prediction[1].

CAM takes a slightly different approach, but produces a similar result. CAM takes the last fully connected layer from a CNN, and takes a weighted average of the last layer's activations to produce a heatmap showing the most important parts of the image [6]. Grad-CAM builds on this by backpropagating the output to the last convolutional layer and applying a similar concept, allowing CAM to be applied to more general neural networks[5].

One common aspect of all such available methods to understand the weights of image classification models is that they generate images for human to interpret. In other words, they show the relationship between the image and its class by presenting visual representation of the weights.

3. Experiment Design

Whereas word vectors such as Word2Vec has been a common method to understand the words in Deep Learning, it was common to visualize the weights as images to understand image classification tasks. To explore a novel way for understanding images, we designed Im2Vec, a language model approach to visualizing and understanding image classifcation. Here, the pixel values of the images are used in place of the context words used for Word2Vec. The main hypothesis behind the design is that just as Word2Vec word embeddings show relationship and similarity among words, the Im2Vec image embeddings will show similarity among visually similar objects.

3.1. Model Design

A common aspect between the skip-gram model used for Word2Vec and image classification is the presence of one-hot encoded vectors. However, image classifciation has the one-hot vectors as outputs, unlike the skip-gram model that takes them as inputs. Therefore, we cannot naïvely take the ith column or the row corresponding to the one-hot encoded item as it is done in the skip-gram model. In our design, to preserve the idea that the result of the matrix multiplication of the weight and the one-hot vector is the word vector, we multiply the output one-hot vector with the inverse of the weight matrix. This adds a constraint on the size of the weight, forcing it to be a $n * n$ matrix where n is the number of classes. The model being trained is simple: it is a 2 layer neural net with just soft max function before the output layer and with no activation functions in between. The reason the model is two layer, not just one, is because the weight matrix that would be used to get the embedding vector should be a square matrix.

Upon deriving the image embedding vectors, we apply Principal Component Analysis(PCA) to reduce the vectors into 2 dimensions in order to visualize it. When visualizing we produced two different versions, one un-normalized and the other normalized.

3.2. Dataset

We have chosen CIFAR-10 and CIFAR-100 to evaluate our hypothesis. These have been selected because of their relatively small input size(32×32) and the small size of the entire dataset. As the dimension of the weight is constrained by the number of target classes, we found it inappropriate to use datasets with high resolution images such

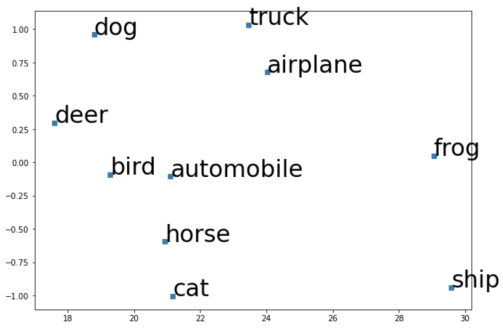


Figure 2: Un-normalized image embeddings for CIFAR-10. The vectors have been reduced from 10 to 2 dimension for the purpose of visualization.

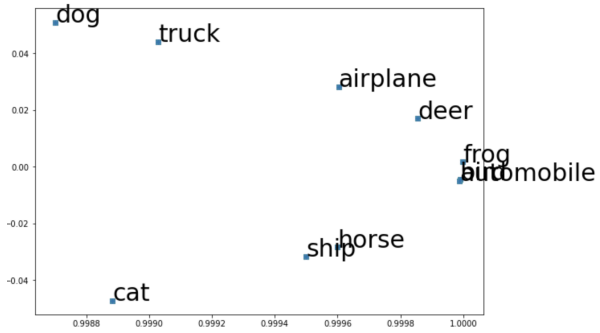


Figure 3: Normalized image embeddings for CIFAR-10. The vectors have been reduced from 10 to 2 dimension for the purpose of visualization.

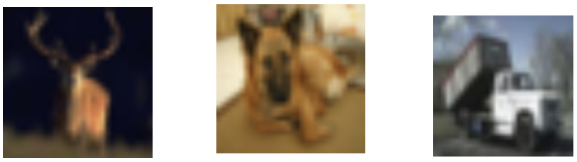
as ImageNet, which would make the task too complex for our shallow model. In particular for CIFAR-100, as training a small model becomes harder with increasing complexity of the task, we trained the model on each category of the classes separately. For example, suppose 'large omnivores and herbivores' is selected. Then only the images with target classes in the category (camel, cattle, chimpanzee, elephant, kangaroo) is included in the training batch.

4. Results

In this section we evaluate the effectiveness of using Im2Vec by visualizing the image embedding from CIFAR-10 and CIFAR-100.

4.1. CIFAR-10

The model was trained over 500 epochs, reaching around 41% accuracy before extracting the weights. Figure 2 shows un-normalized embeddings from CIFAR-10. From the un-normalized result, it can be said that images of deer and dog are similar in their pixel value. The images of the two classes look quite alike. However, because these are just



(a) Deer (b) Dog (c) Truck

Figure 4: Images from CIFAR-10

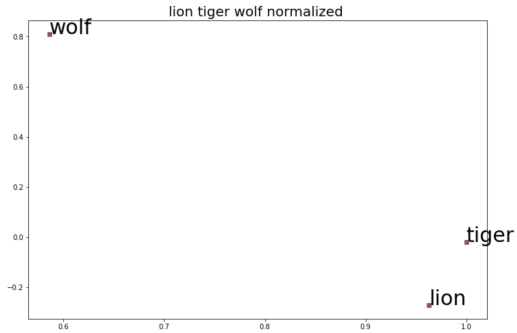


Figure 5: Normalized image embeddings for large carnivores from CIFAR-100. The vectors have been reduced from 100 to 2 dimension for the purpose of visualization.

(a) Lion (b) Tiger (c) Wolf

Figure 6: Large Carnivore Images from CIFAR-100

two of the many images in the dataset, it is questionable whether this conclusion can be generalized. In fact, in the normalized result, truck seems to be the closest class to dog. However, when we compare the two images, the two look quite disparate. Moreover, the normalized embeddings for bird and automobile almost completely overlap, but it is hard to see the two classes as having similar visual representation. As such, for CIFAR-10 Im2Vec seems to be a limited approach to understand images.

4.2. CIFAR-100

For CIFAR-100, we have trained the model over 1000 epochs, considering greater complexity of thet ask compared to CIFAR-10. The average max accuracy over each category was around 10%. This accuracy was achieved even after running just about 100 epochs, but to determine whether this is the upperbound for the training accuracy with the given architecture, we have trained up to 1000

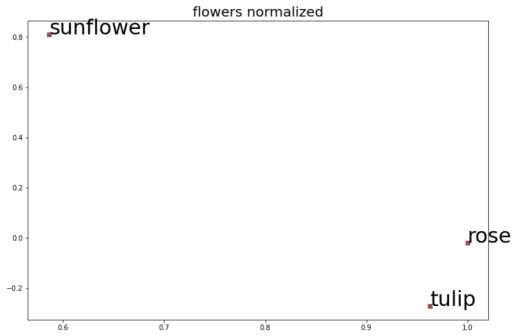


Figure 7: Normalized image embeddings for Flowers from CIFAR-100.

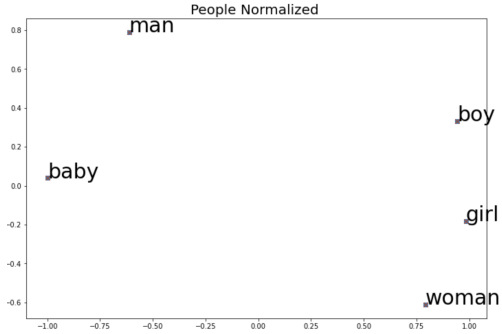


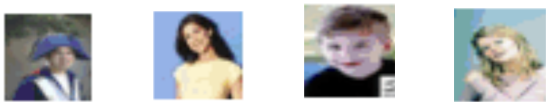
Figure 8: Normalized image embeddings for People from CIFAR-100.

epochs. Even though the achieved accuracy was lower for this dataset, the relationship among different images was more visible, since we focused on a few selected classes by training each category separately.

Figure 5 shows the normalized image embedding vectors for large carnivores–lion,tiger, and wolf. The distance between lion and tiger is significantly less than that between wolf and the other two. Visually, lion and tiger look alike more than wolf and the other two do as well.

Visual similarity manifest in the image embeddings is also present among flower images. The image embeddings extracted from the model after being trained on flowers data are shown in Figure 7. Here the embedding vectors for tulip and rose are closely related to each other. It is true that tulips and roses have similar rounded shape while sunflowers have their distinct spiky shape.

Although Im2Vec shows some relationship among pixel values of different classes, the embedded relationship differs a bit from that of Word2Vec. In Word2Vec, an analogy of women is to man girl is to boy can be clearly shown. However, it is not necessarily true that the pixel values of the corresponding images have the same relationship. The image embedding vectors for woman and girl are adjacent



(a) Man (b) Woman (c) boy (d) girl

Figure 9: People Images from CIFAR-100

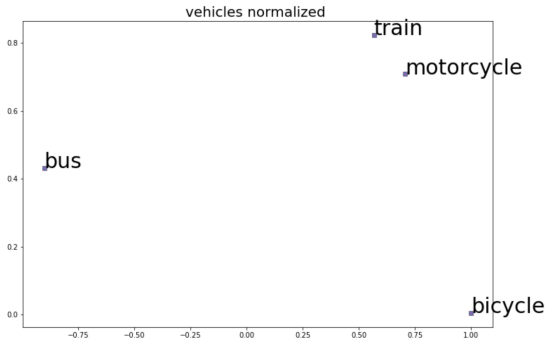


Figure 10: Normalized image embeddings for Vehicles from CIFAR-100. In this case Im2Vec fails to reflect visual relation among images

to each other, whereas those of man and boy are quite far from each other. The vector arithmetic $woman - man = girl - boy$ cannot be applied here. However, the vectors can be understood visually by looking at the images. Figure 9 shows randomly selected images for each class of the people category. Woman and girl are visually similar to one another and so does boy and girl. However, the image for boy and man are not as close as it is for the others. This visual relation is reflected on the normalized image embeddings as shown in Figure 8.

Im2Vec effectively encodes some visual relationship in the vector embeddings, this may not always be the case. When we have tested the vehicles category, the embedding vectors were much different from our expectation. Intuitively, we have expected the motorcycle to be close to bicycle, and train to bus. However, the train and motorcycle vectors are clustered together. As it is the case, Im2Vec may not be an effective approach to understand images in some cases.

5. Conclusion

In this work we presented Im2Vec, a novel approach to understanding the images in image classification task. Whereas conventional methods have tried to generate images from the trained model, we sought to derive a numerical representation: image embedding vectors. The main idea for Im2Vec is using the pixel values of each image as the context for each its classification. By doing so, the generated image embedding vectors can reflect the relative

importance of each pixel for each class as well as show relationship between different classes via the proximity between their vectors.

Overall, the results for Im2Vec were mixed, with clear successes in some classifications, and failures in others. Im2Vec failed to produce useful representation for CIFAR-10 dataset and select categories from CIFAR-100. However, for many categories for CIFAR-100, visual similarity of different classes were reflected on image embedding vectors, as they were clustered together.

Future works include improving Im2Vec by lifting the constraint set on the size of the weight matrices by using pseudo-inverse instead of regular inverse matrices to generate image embedding vectors. In addition, approaches such as GAN, VAE, and deep neural network, which deviate from the origianl Word2Vec design, can be explored as a new way to generate image embedding vectors that capture relationship among various classes. To make the image embedding vectors more robust to variances in images, intense data augmentation can be applied as well.

References

[1] Chinesh Doshi. Generalized way of interpreting cnns!!, Aug 2020.

[2] Chris McCormick. Word2vec tutorial - the skip-gram model, Apr 2016.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[5] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[6] B. Zhou, A. Khosla, Lapedriza. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.

A. Appendices

A.1. Model

We used a two layer neural network which took a flattened image as the input layer, and with a hidden layer having as many neurons as there are image classes. For CIFAR-10 this was 10, and for CIFAR-100 this was 100. The last layer was softmax with cross entropy loss.

A.2. Datasets

The model was trained on specific subsets of the CIFAR-10 and CIFAR-100 datasets available at https://www.cs.toronto.edu/ kriz/cifar.html .

A.3. Training

The CIFAR 10 dataset was trained over 500 epochs using the SGD optimizer with parameters 0.01 weight decay, 0.2 momentum, 0.01 learning rate and 512 batch size. The learning rate was optimized by decreasing it until the loss converged over many iterations. Momentum and weight decay were chosen by using similar values from other simple models trained on CIFAR. After seeing that the accuracy was too low after 300 epochs, we increased the number of epochs trained to 500 for CIFAR 10, and 1000 for CIFAR 100. Runtimes were aroundn 4 hours for 1000 epochs for CIFAR 100 on a Titan Xp GPU with 8 workers running on Seoul National University computing clusters.