Semantic-aware Image Similarity Search

1st Yueming Zhang
SUNET ID 06487360
Stanford CS230
Vancouver, BC, Canada
mzcanada@stanford.edu

***Abstract*—This paper seeks improvement on image similarity search. In addition to support a query image as the only input parameter, we will also support additional query attributes as combined search criteria. These additional attributes include supplemental text attributes and image native attributes. Supplemental text attributes are image's title, caption or category, as a natural target of NLP. Image native attributes are attributes embedded in the image itself such as color. Combining text and image attributes injects semantic meaning from NLP to an image, enables composite search and semantic arithmetic operation on image data. Our result shows 84% accuracy of image search and 79% accuracy of image arithmetic operation.**

I. INTRODUCTION

Convolution Neural Network (CNN) advanced reverse Image Search in recent years. The implementation is typically two steps: (a) use CNN as feature extractor to create image embedding database, and (b) use nearest neighbour algorithm to perform approximate matching. Such search accepts one image as the query image, and return multiple images as the output. This type of search doesn't support combined filter, i.e. query by an image and a supplement text attribute as an additional refinement filter. Able to interactively supplement a text attribute on a stock photo site, or a sub-category attribute on an e-commerce site are critical for user experience. Concretely, this paper has two goals:
(1) Create semantic aware composite embedding: combine image feature, natural language feature (GloVe), and other image native features (multiple colors of an image)
(2) Enable interactive search: A mechanism to query such model with query-time adjustable filters, and support sub-second response time on million-scale images
These two goals are tightly interrelated: the information stored in the composite embedding space must be sensitive to the query weight of each search criteria. These two goals act together to full-fill below use cases:

- a user upload a query picture with mountains at the background to perform similarity search. Because no other query criteria, the algorithm assigns 100% weight to the picture, and 0% to other attributes.
- After seeing the returned pictures, the user decides to adding 'vehicle' to the foreground, so he supplements the text "vehicle" as an additional query criterion
- The algorithm now balance the weight between the image and text 'vehicle', and return pictures with both elements. After seeing the new images, the user may increase or decrease the importance of 'vehicle' in the picture by adjusting the weight. The result is as shown in Fig [6].
- Then the user may only want to have 'car', so he replace the text 'vehicle' with 'car'. However, there is no 'car' in any picture has a mountain background. The algorithm then leverage semantic meaning of 'car', return semantically closer pictures, i.e. a mountain with 'truck', rather than a mountain with 'flower', because 'truck' is semantically closer to 'car' than 'flower'.

Clearly, the network needs to learn from both picture content, and picture title, and bridge the gap, i.e. even a picture's title doesn't contain 'car', the network needs to learn from the content if there is a car.

II. RELATED WORK

NLP and Image Embedding. NLP embedding has been thoroughly studied in recent years. Language vector representation in latent space demonstrated outstanding performance of linear arithmetic semantic relationship (king-queen=man-woman) in various language model such as word2vec and GloVe as discussed in [5] and [6]. These NLP-based latent space is surprisingly similar to the CNN-based image embedding in computer vision. Both fields rely heavily on creating a high-dimensional embedding as the unique representation for images and words. Both fields have pre-trained models that can be used to extract these embedding, such as VGG16/ResNet for image, and GloVe/Bert for NLP. One key difference between language embedding and image embedding is that, the former is semantic-aware, while the later isn't. For example, in NLP embedding space, the distance between a car and a road is closer than the distance between the a boat and the road. But in image embedding space, a picture with a car may not be closer to a picture with road than a picture with ocean.

Bridge the gap between image and NLP. In addition, recent studies on image and text correlation resulted plenty of papers about image/text two way translation: image captioning, and search image by text. Among which are reliable image classification [1], image description generation [2], image annotation and search [3], attention-guided multi-modal search [4], and more. To bridge the gap between image and NLP, many of these current papers determine the optimal weight distribution at training time, and yield hard-coded weights that are not adjustable at query time. These studies can not be

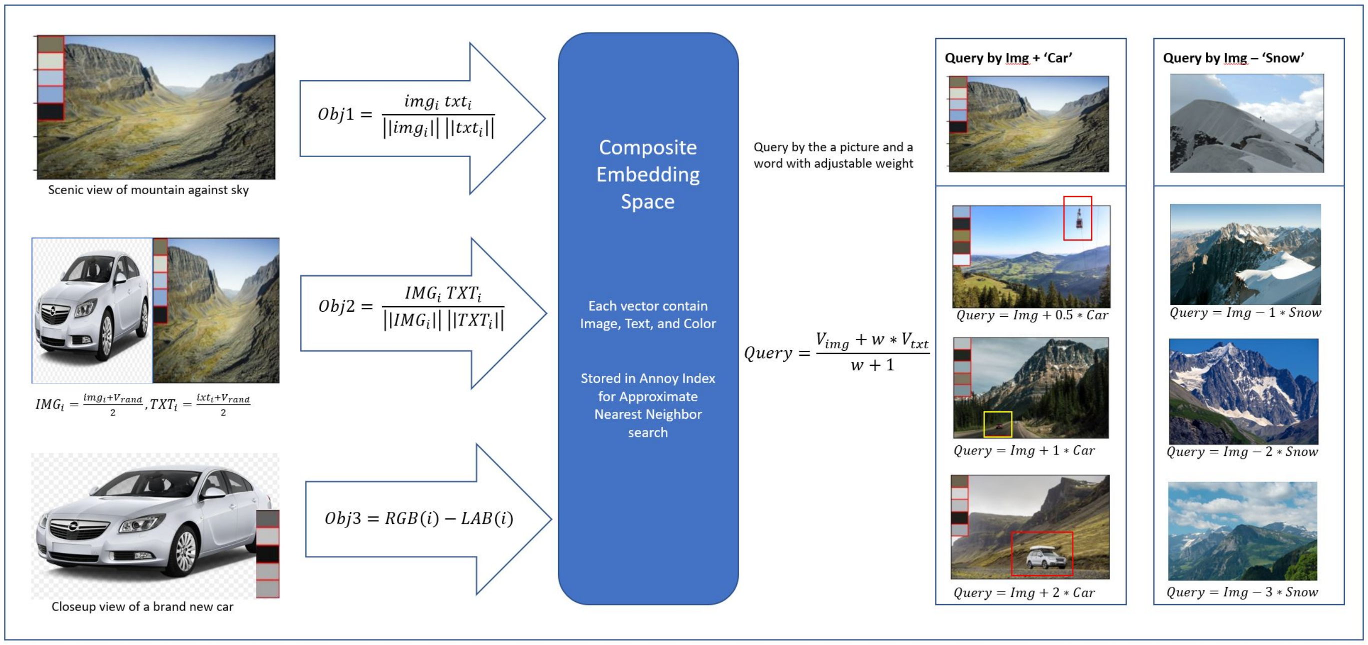

Scenic view of mountain against sky
$Obj1 = \frac{img_i\ txt_i}{||img_i||\ ||txt_i||}$
$IMG_i = \frac{img_i + V_{rand}}{2}, TXT_i = \frac{txt_i + V_{rand}}{2}$
$Obj2 = \frac{IMG_i\ TXT_i}{||IMG_i||\ ||TXT_i||}$
Closeup view of a brand new car
$Obj3 = RGB(i) - LAB(i)$
Composite Embedding Space
Each vector contain Image, Text, and Color
Stored in Annoy Index for Approximate Nearest Neighbor search
Query by the a picture and a word with adjustable weight
$Query = \frac{V_{img} + w * V_{txt}}{w+1}$
Query by Img + 'Car'
$Query = Img + 0.5 * Car$
$Query = Img + 1 * Car$
$Query = Img + 2 * Car$
Query by Img – 'Snow'
$Query = Img - 1 * Snow$
$Query = Img - 2 * Snow$
$Query = Img - 3 * Snow$

Fig. 1. **Left**: create Composite Embedding based on 3 objectives. Objective 1: reduce the cosine distance between each picture's feature vector (img_i) and the picture label's NLP vector; Objective 2: reduce the cosine distance between the two vectors:IMG_i and TXT_i, which are generated through linear arithmetic operation. and 3: mapping RGB to LAB color space. **Middle**: store embedding in Annoy Index for fast approximate nearest neighbour search. **Right**: retrieval by image and a text with adjustable weight, observing increasing weight of 'car' or decreasing weight of 'snow'. Also note a 'gondola' shown up because of the closer semantic distance to a 'car'

directly used to meet the goal of this paper, but serve as a solid foundation to advance image similarity search.

III. APPROACH

There are three parts of work to enable end-to-end interactive search experience: 1. train a Projector Model to create Composite Embedding. 2. create color extractor to determine 5 colors from each picture and append to the Composite Embedding, and 3. storage and in-memory presentation of the embedding that enables query with adjustable weights and sub-second response time on million-scales images.

We leverage pre-trained CNN model, Efficient Net B3(EFN3), as the backbone to extract features from images. Although many pre-trained models such as VGG, ResNet, Inception, etc. are available, EFN3 is at a sweet spot balancing between search accuracy and computational cost. The feature embedding created by Efficient Net (and other models) do not have semantic meaning. The primary goal of the Projector model is to create a one way mapping from image embedding to NPL embedding so that the closer images are semantically relevant. We will discuss experiments that lead to the choice of the final model architecture and the end to end solution.

A. Experiments

The first attempt (baseline) is to use a sequential model to map image embedding to NLP embedding (generated from caption/label of the image). The feature output from EFN3 is used as the image embedding, which is the input of our Projector Model. Output of the Projector Model is the projected embedding. The objective is to minimize the distance between the projected embedding and the NLP embedding. This approach works but has two major issues:

- EFN3 is an image classification model. The goal of this model is to accurately classify the most dominant object in the image. The extracted feature also sufficiently represent this most dominant object so the image can be classified correctly. As a result, the feature of less dominant objects may not be sufficiently represented. This misaligned with our goal as NLP embedding often contain more information than the dominant object as shown in Fig (4), i.e. 'Boy Standing Against Trees'. The information of less dominant object and the localization awareness might be lost before reaching the last layer of EFN3.
- To solve above issue, we need to extract information from earlier layers of EFN3. Making all layers of EFN3 trainable could guide the network to focus on relevant image patches. However, we found out that training the whole network is too computationally expensive to be a viable solution.

After solve above two issues, ideally minimizing the distance between the projected embedding and NLP embedding should be sufficient to create the semantic linkage. However, our further experiments showed that image retrieval is accurate, but not arithmetic (image + text) offset retrieval as shown in Fig [5]. Our proposed model address issues discovered from these experiments.

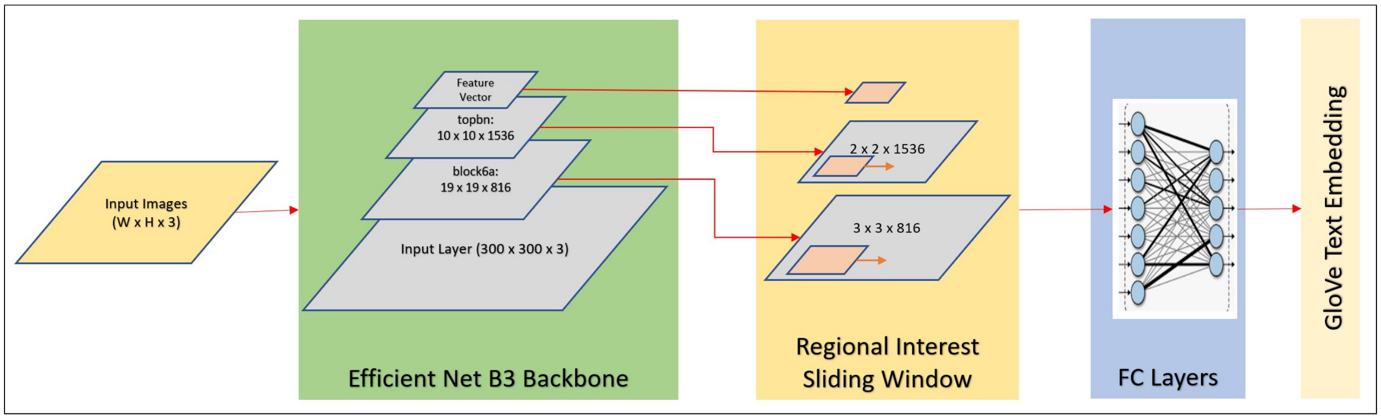


Fig. 2. Our Projector Model. **Backbone**: EFN3 is used as the feature extractor. In addition of extracting feature vector from the final layer, the output of two middle layers are also extracted and feed into the regional sliding window. **Regional Interest Sliding Window**: we apply Average Pooling to create 14 regions from the later layers to maintain regional awareness. These 14 regions are: one region for the whole image, 4 regions from top batch normalization layer, and 9 regions from earlier block6 layer. **FC Layers** is a sequential model to map GloVe Text Embedding.

B. The Model

Our Projector model is defined in Fig [2]. To keep the regional awareness, we extract output from two middle layers of the backbone, and then split images in the two layers into 4 tiles and 9 tiles, respectively. It's common for object detection tasks to run sliding window in more fine-grained steps and on more layers to identify each bounding box location. Our goal is to correlate image content with text description, thus, bounding box is not required. The 14 tiles (aforementioned 13, and the feature of the whole image) are sufficient to learn such linkage.

The two objectives defined in Fig [1] leads to following loss function:

$$\begin{aligned} Loss_i &= \cos(img_i, txt_i) + \cos(IMG_i, TXT_i) \\ txt_i &= \frac{\sum_{j=1}^{n} V_j}{n} \\ IMG_i &= img_i + V_{rand}, \;\; TXT_i = txt_i + V_{rand} \end{aligned} \tag{1}$$

The first part of the loss function is to meet Objective 1: map an image vector to text vector. The second part is used to improve the accuracy of the second objective, so that we can add (or remove) a word with adjustable weight from a picture. V_j denotes the embedding of j^{th} token in the image label (length = n). The V_j is extracted from each token in the image label using GloVe dictionary. txt_i is the mean of all vectors of words in the label. $IMG_i = img_i + V_{rand}$, is the summation of an image's feature vector (img_i) and a random word's NLP vector (V_{rand}). $TXT_i = txt_i + V_{rand}$, is the summation of the label's NLP vector (txt_i) and the random word's NLP vector;

C. Noisy Label

Since the Projector model bridge the gap between image and text, we would expect the new embedding space has clear semantic meaning. However, because noisy image label contains various type and form of words, it requires preprocessing before calculate the embedding. We use nltk WordNet package performed two steps after tokenization: (1) remove stopping words, i.e. a, the, this, myself, etc. (2) word stemming to convert word into it's base form. We also evaluated Part-Of-Speech filer and word frequency threshold filter, but found these two filters, although can greatly improve accuracy, will negatively impacting search experience. We skipped the last two filters.

D. Image Color

Image colors need to be part of the composite embedding to make the color search-able. We initially attempted K-Means to extract 5 RGB colors, but found out K-means is too computational expensive. We later switched to a sampling approach that 100 times faster with acceptable performance. The distance between two RGB colors don't proportionally reflect human perception, for example, human commonly perceive RGB below 30 as black and not sensitive to the color changes at this range. We converted RGB to LAB color space to align with human perception.

E. The query with dynamic weight

We use approximate nearest neighbour (ANN) search open source package (Annoy) to achieve sub-second response time on million-scale data. ANN search takes one input query vector, and returns multiple vectors with distance in ascending order. This caused two challenges:

(1) it doesn't support conditional parameter. When a user only query by image (without any colors), we need only query the image and ignore the color embedding. One option is to create two embedding: image embedding and color embedding. We search the image embedding first and then narrow down the result with color filter. However, this approach significantly slow down the query, because the image query needs to return a large number of records as the input of the secondary color search.

(2) one picture contains multiple colors, a query needs to return the image if one color is close enough to the query color. Unfortunately no ANN algorithm supports queries like:

if color1 == 'red' or color2 == 'red', etc.

We duplicate each image's feature vector into five vectors with identical image embedding, but different color embedding. This supports efficient color search, enable us to use below query to adjust weight before passing into the ANN search.

$$Query = (1 - p) \times V_{query} + p \times V_{color} \qquad (2)$$

$$V_{query} = \frac{V_{img} + V_{txt} * w}{w + 1}$$

The equation support two query-time adjustable parameters: w and p, where $w \in [-\infty, \infty]$ denotes the weight of the arithmetic word, $p \in [0, 1]$ denotes the weight of input color. This makes color, text, and image itself all optional: When w is large, the algorithm ignores the query picture content and use the text as the only criteria.

F. Training

It's very computational expensive if we include EFN3 in the training pipeline. Since we don't plan to train EFN3 parameters, there is no need to run images through EFN3 for every forward propagation. We run all images through EFN3 once, and store the output (as shown in Fig [2] in a database. This drastically speed up the overall training process. We use Adam optimizer, batch size 64, and use callback to log training metrics. The accuracy calculation can't be done within the single mini-batch because of the arithmetic operation. We again rely on Keras callback at the end of each epoch to perform custom accuracy calculation.

IV. Dataset

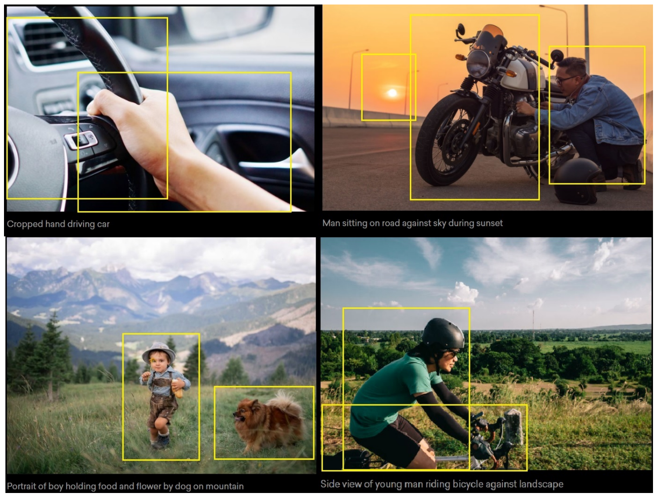
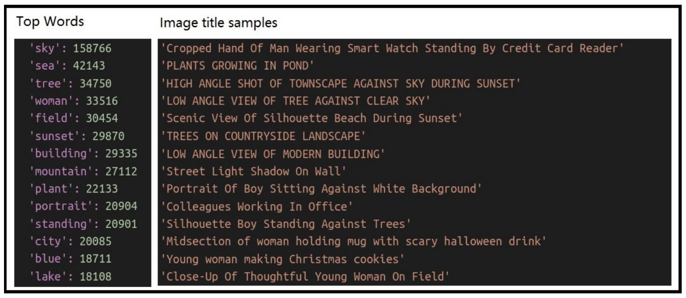

Cropped hand driving car
Man sitting on road against sky during sunset
Portrait of boy holding food and flower by dog on mountain
Side view of young man riding bicycle against landscape

Fig. 3. Each image has one label contains multiple words. This labels describe the content of a image. Typically a label describes two or three dominant objects at different region of a picture

The experiments are based on 100,000 pictures of a stock photo website. These pictures are in compressed .jpg format with resolution typically around 1000 * 1000 pixels. Each pixel has RGB value between 0-255.
Each image has one label contains multiple words, as shown in Fig [3]. These labels typically describe two or three most dominant element in the picture, i.e. 'Cropped hand driving car', or 'Young woman riding bicycle against landscape'. Simple classification of such image will reveal correlation between the description of the most dominant element and the most dominant element itself. However, since each label typically describe two or three elements in different regions, it's not effective to simply using image classification backbone.

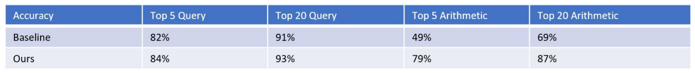


Top Words	Image title samples
'sky': 158766	'Cropped Hand Of Man Wearing Smart Watch Standing By Credit Card Reader'
'sea': 42143	'PLANTS GROWING IN POND'
'tree': 34750	'HIGH ANGLE SHOT OF TOWNSCAPE AGAINST SKY DURING SUNSET'
'woman': 33516	'LOW ANGLE VIEW OF TREE AGAINST CLEAR SKY'
'field': 30454	'Scenic View Of Silhouette Beach During Sunset'
'sunset': 29870	'TREES ON COUNTRYSIDE LANDSCAPE'
'building': 29335	'LOW ANGLE VIEW OF MODERN BUILDING'
'mountain': 27112	'Street Light Shadow On Wall'
'plant': 22133	'Portrait Of Boy Sitting Against White Background'
'portrait': 20904	'Colleagues Working In Office'
'standing': 20901	'Silhouette Boy Standing Against Trees'
'city': 20085	'Midsection of woman holding mug with scary halloween drink'
'blue': 18711	'Young woman making Christmas cookies'
'lake': 18108	'Close-Up Of Thoughtful Young Woman On Field'

Fig. 4. Each image has one label contains multiple words. There are total 17,622 unique words after word cleansing, and in average 7 words per image. The 1st column, Top Words, shows some most used words in the dataset

Each label typically contains 3 to 20 words. The average length of all labels is 7 words. Labels are noisy in two perspectives: a. may not have clear association with a specific element in the picture, for example the top right picture in Fig [3] has a label 'Man sitting on road against sky during sunset', which omitted the motorcycle. and b. labels contain words without a clear visual indication, such as 'On', 'View', 'Against', etc. as shown in Fig [4].

V. Evaluation Metrics



Accuracy	Top 5 Query	Top 20 Query	Top 5 Arithmetic	Top 20 Arithmetic
Baseline	82%	91%	49%	69%
Ours	84%	93%	79%	87%

Fig. 5. Our model outperformed baseline on both Image retrieval (Obj1), and Image Arithmetical (image + text) retrieval (Obj2) accuracy. The image arithmetical linear retrieval, (such as a Mountain Picture + a word 'Car'), outperformed baseline by a substantial margin

We evaluate the two metrics that align with the first two objectives: Single Image Search Accuracy Obj_1, and Image + Text composite search Accuracy Obj_2. Img_n denotes the top n results (query by image) from composite embedding space. Txt_n denotes the top n results (query by label) from the native NLP embedding space (GloVe). We define query score as

$$f(i) = \begin{cases} 0, & \text{if } Img_n \cap Txt_n = \varnothing \\ 1, & \text{otherwise} \end{cases} \qquad (3)$$

here $f(i)$ is 1 if query by the image's vector and query by the image label's vector yield overlapping result. Finally the accuracy of Objective 1 is defined as:

$$Obj_1 = \frac{\sum_{i=1}^{m} f(i)}{m}$$

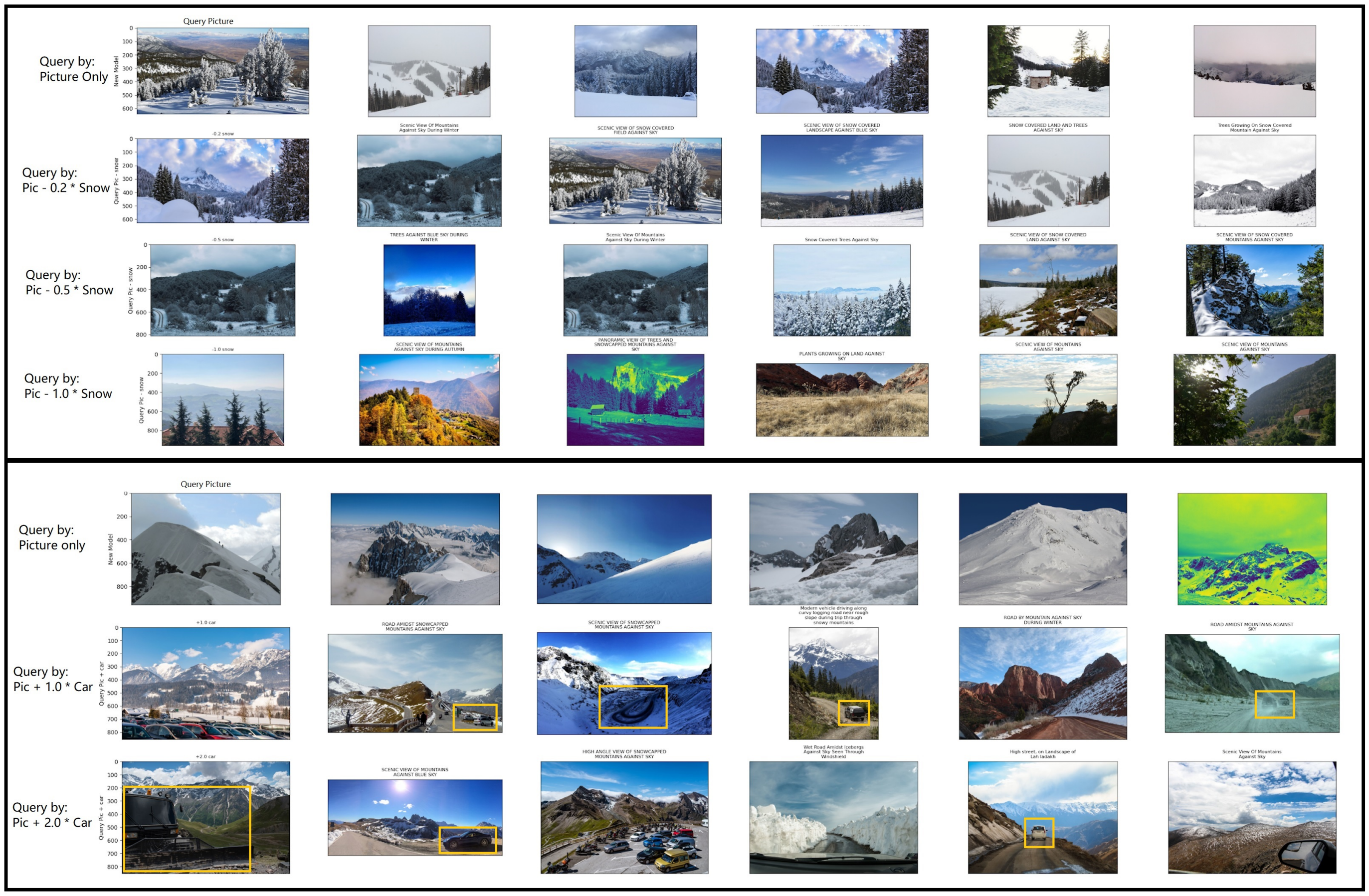

Query Picture
Query by: Picture Only
Query by: Pic - 0.2 * Snow
Query by: Pic - 0.5 * Snow
Query by: Pic - 1.0 * Snow
Query by: Picture only
Query by: Pic + 1.0 * Car
Query by: Pic + 2.0 * Car

Fig. 6. Qualitative Evaluation. **Top: Remove Element From a Picture**: Query by a picture of snow-covered mountain range returned similar pictures at the top row. Then we gradually remove snow by a weight of 0.2, 0.5 and 1.0. $2^{nd}, 3^{rd}$ and 4^{th} rows clearly show that the retreat of snow, while still maintain the characteristic of the original query picture: mountain. **Bottom: Add Element to a Picture** We query by a picture of snow-capped mountain peak, and result is shown in the first row. Then we add an element of Car to the picture by weight of 1.0 and 2.0, resulted in the 2^{nd} and 3^{rd} rows. Observing the weight increase of car from 2^{nd} to the 3^{rd}. **Two interesting outcomes:** a) 2^{nd} tile in the 2^{nd} row shows multiple cars, while the title contains nothing related to car, indicating the netwok learned from the picture content, and b) 3^{rd} tile in the 2^{nd} row shows a road, demonstrating the semantic awareness of the model as a road has closer distance to a car

Accuracy of Objective 2, Obj_2, is defined similarly except adding a random word's vector to the query image's vector, and add the same vector to the image label's vector. The random natural also introduced a helpful regularization effect allowing train longer.

Baseline result is measured from a sequential model that minimize the cosine distance between image vector and text vector (equivalent to Obj1(Fig[1]) alone), without middle layer feature extraction. Since retrieval typically returns multiples images, we evaluated both top 5 and top 20 accuracy.

Quantitative evaluation result is described in Fig [5]. The top 5 query accuracy and top 5 arithmetical accuracy reached 84% and 79% respectively. Qualitative evaluation result is described in Fig [6], demonstrated the semantic awareness of the new embedding space.

VI. Error Analysis

Analyzing the errors revealed two patterns. 1) Accuracy decreases as the sentence gets longer. We use the mathematical mean of each word's GloVe vector as the target embedding. This approach is less effective when there are lots of words in the sentence. 2) Accuracy decreases for the words appear at lower frequency. The training sample size of less used words is not sufficient for the network to learn the pattern.

VII. Conclusion

This paper explored ways to bridge image embedding with language (GloVe) embedding to support Image + Text composite search with adjustable weight. Images in the new embedding space have clear semantic meaning that support linear arithmetic query, as shown in Fig [6]. Our method's unsupervised nature, and the effectiveness on noisy data (downloaded from a website), demonstrated the potential to replicate such approach in a multi-tenant environment. Future research can replacing GloVe with BERT to support longer sentences with attention mechanisms.

References

[1] Devise: A deep visual-semantic embedding model. Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et. al

[2] Deep visual-semantic alignments for generating image descriptions. Andrej Karpathy and Li Fei-Fei.

[3] Associating neural word embeddings with deep image representations using fisher vectors. Benjamin Klein, Guy Lev, Gil Sadeh, and Lior Wolf.

[4] Attention guided Multi-modal Correlation Learning for Image Search. Kan Chen, Trung Bui, et. al.

[5] Unifying Visual-Semantic Embeddings with multi-modal Neural Language Models. Ryan Kiros, Ruslan etl.

[6] Linguistic regularities in continuous space word representations. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig.

[7] Feature extraction on large dataset with Deep Learning: https://www.pyimagesearch.com/2019/05/27/keras-feature-extraction-on-large-datasets-with-deep-learning/

[8] EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks: https://arxiv.org/abs/1905.11946

[9] The Ultimate Guide to 12 Dimensionality Reduction Techniques: https://www.analyticsvidhya.com/blog/2018/08/dimensionality-reduction-techniques-python/

[10] Approximate Nearest Neighbors by AnnoyIndex: https://github.com/spotify/annoy

[11] Ann Benchmarks.com: http://ann-benchmarks.com/

[12] DeViSE: A Deep Visual-Semantic Embedding Model (Google, afrome, et. al)

[13] Joint Visual-Textual Embedding for Multimodal Style Search (Amazon, Gil Sadeh, et. al)

[14] Efficient Large-Scale Multi-Modal Classification (Facebook: dkiela, et. al)

[15] Face recognition: A literature survey. Zhao, W.; Chellappa, R.; Phillips, P. J.; and Rosenfeld, A.

[16] Cnn features off-the-shelf: an astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Razavian, A. S.; Azizpour, H.; Sullivan, J.; and Carlsson

[17] Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng.

[18] A multiplicative model for learning distributed text-based attribute representations. Ryan Kiros, Richard S Zemel, and Ruslan Salakhutdinov.