Localized Embeddings (LocEm): A multi-task model for localization, classification, and retrieval of multiple objects in images

Shravan Kale
Computer and Information Science
University of Oregon
shravank@cs.uoregon.edu

Abstract

Embeddings are generated across different Deep Learning models to represent objects or entire images. They are then used for tasks such as Object Retrieval by matching against a database of other object embeddings. We propose our model LocEm, a single passthrough model that can generate embeddings for multiple objects in images but at the object's location. We also repurpose the ImageNet video dataset that includes natural augmentation containing pose and action movement variation of objects in images to create a triplet generator. Our method to create LocEm includes extending an existing object detection model with the ability to predict embeddings and include an extended triplet loss function that encodes embeddings generated for non-object (background) entities in an image. We evaluate our model against other models in the literature that at most perform two of the three tasks performed by our model, i.e., object detection, classification, and embedding generation. The models are evaluated for fine-grained categorization based on object retrieval. We keep the base architecture and dataset constant across the models to show that the strength of our model is derived through the final LocEm layers and our training loss function. We also discuss the performance of our model on the variety of object instances and methods for further improvement.

1. Introduction

In Computer Vision, object retrieval is a fundamental task that has benefitted from the advent of Deep Learning. The task consists of building a database of object representations and using query image representations to identify objects. The said identification enables retrieving identical objects, fine-categorization of objects, or identifying variations of the query object. A common representation of an object is an embedding of any d dimension. The numerical values in the dimensions can be hand-crafted or learned, and the embedding is used for similarity matching with a database of embeddings.

The task of object localization, classification, and embedding generation is either performed using separate network models or in a combination of two tasks. The goal of this work is to create a single-pass through model that is able to classify and localize the embeddings of object instances. Towards this effort, we have extended the Yolo [18] model to predict embeddings localized to the object region in an image. To evaluate the embeddings generated by our model we apply them in the task of fine-grained categorization. Object retireval results can be used as a heuristic for narrowing the search space for matching embeddings yielding the finer categorization. In the following sections, we have described the details of our current work. The Section 2 discusses the related work including the transition from machine learning based embeddings to the newer deep learning based embeddings. In Section 3 we discuss the dataset and its modifications, Section 4 details the methodology for LocEm, Section 5 details the experiments performed for the task at hand, and Section 6 tabulates the results of the experiments. The Section 8 discusses the entire loss function used for LocEm.

Our contributions are as follows:

1. A triplet generator for a dataset of objects extracted from videos. The objects encode natural augmentations such as pose and action movement variations.
2. A single passthrough multi-task model for localizing embeddings for multiple objects in an image
3. A modified triplet loss function that proposes the use of background data (embeddings) as negative pairs for the objects in an image.

2. Related Work

The general methodology used in object retrieval consists of image feature engineering followed by efficient indexing and matching using query objects in images. The

initial advancements in object retrieval or Content-Based Image Retrieval were led by the introduction of SIFT [10] descriptors that generated discriminative representations with invariance to transformations such as rotations and scaling. The inspiration for object retrieval was drawn from text retrieval [21], they compute SIFT descriptors over multiple regions of stable object frames of movie videos and use them as object descriptors (embeddings). The descriptors of these regions or visual words can be arranged and indexed using different methods to improve the query search aspect of retrieval systems.

Descriptors can also be generated using randomized trees [15] and approximate nearest methods for a scalable vocabulary of visual words. The indexing of descriptors is achieved using an inverted file system data structure [15], [21]. The embedding generation method (quantization) and indexing are combined in a vocabulary tree structure [12] where hierarchical TF-IDF is used for weighting individual descriptors. The matching of descriptors can be improved by using the probabilistic relationship between the visual words [11] compared to the previously used L2 matching [15], [12]. Another approach to matching consists of using hamming embeddings [6] (a matching technique) and weak geometric consistency [6] for filtering the retrieval matches with inconsistent angles and scales of object.

Post-processing techniques are employed to further improve descriptors such as spatial verification [15] to re-rank the retrieval results. Query expansion [3], [23] is employed by reusing relevant and retrieved (matched) objects from the inverted file system to generate an improved descriptor. Other techniques for descriptor improvements include augmentations using feature aggregation [22] and feature selection [25] of descriptors.

In addition to SIFT descriptors, other works included the aggregation of lower level features of objects using a Gaussian mixture model in Fisher Kernels [13], its simplification in VLAD [7], and by using hashing techniques for compression [14]. Descriptor compression techniques using dimensionality reduction [5], [16] have also been studied to reduce the large dimension size of the descriptors.

The success of Convolutional Neural Networks (CNN)[9] on the ImageNet classification task proved its capability to extract state-of-the-art features compared to the previous descriptors techniques [10], [13]. The extracted features are used as a general-purpose representation of images in tasks such as fine-grained image classification [20] and image retrieval [1] where images consist a single object. A CNN [20] is pre-trained on ImageNet dataset, and it's fully connected layers are then used to train a separate Support Vector Machine on different visual tasks. The fully connected layers can be used as descriptors [1] after retraining the pre-trained CNN using the test dataset. The dimensionality of those descriptors is then reduced using Principal Component Analysis (PCA). In contrast to the previous techniques, the last convolutional layer's sum-pooling layer [17] can also be used as a descriptor for image retrieval.

The above methods were yet outperformed by traditional methods that employed geometric verification [15] and query expansion. A CNN's feature extraction ability was further improved with the introduction of R-MAC[24]. It aggregates features by max-pooling over a set of local regions over all the channels from the last convolutional layer. The R-MAC descriptors of images are used to match the query images and the database images, followed by re-ranking and query expansion. A siamese network [4] further improves on R-MAC by retraining the CNN end-to-end with a triplet ranking loss instead of using only the pre-trained convolutional features. It also optimizes the weights of R-MAC descriptor generation and employs a regional proposal network to learn the regions to pool from the siamese network.

Complementary to our work, involves the construction of hard positive and negative image samples from camera viewpoints that exploit the three dimensional model geometry [17]. Similar to R-MAC, it uses a siamese network as its backbone architecture and a contrastive loss to train the network. In contrast to the average pooling layers used in the previous models, Generalized Mean Pooling (GeM) [17] is introduced to generate descriptors over the last layer of the siamese network. The descriptor learning is followed by whitening, PCA, and query expansion.

3. Dataset

The dataset used for this task is the ILSVRC2015 ImageNet Videos dataset [19] created for object detection in videos. The videos are divided into 4417 snippets, which are further divided into a total of 1258K frames or images. The annotations of said images consist of 1002K bounding boxes of 30 object categories, divided between the training and validation sets.

We use the videos dataset because it provides various images of the same object throughout a time-series. It gives us the ability to use multiple instances of the same objects in a triplet as a positive pair required for our task. Given the objects are from videos, we also benefit from the natural augmentations provided by the movement of the objects or its stationery location with a change in stance or view. We reduce the number of triplets generated from the multiple instances of the same object by limiting our selection to two instances of every object. To achieve this, we use a composite key of the dataset, which consists of the category code, snippet id, and the tracking id of objects together identifying every unique object. For every unique composite key, we select two instances at random, generating 15806 objects/triplets from 15486 images for training and 2618 ob-

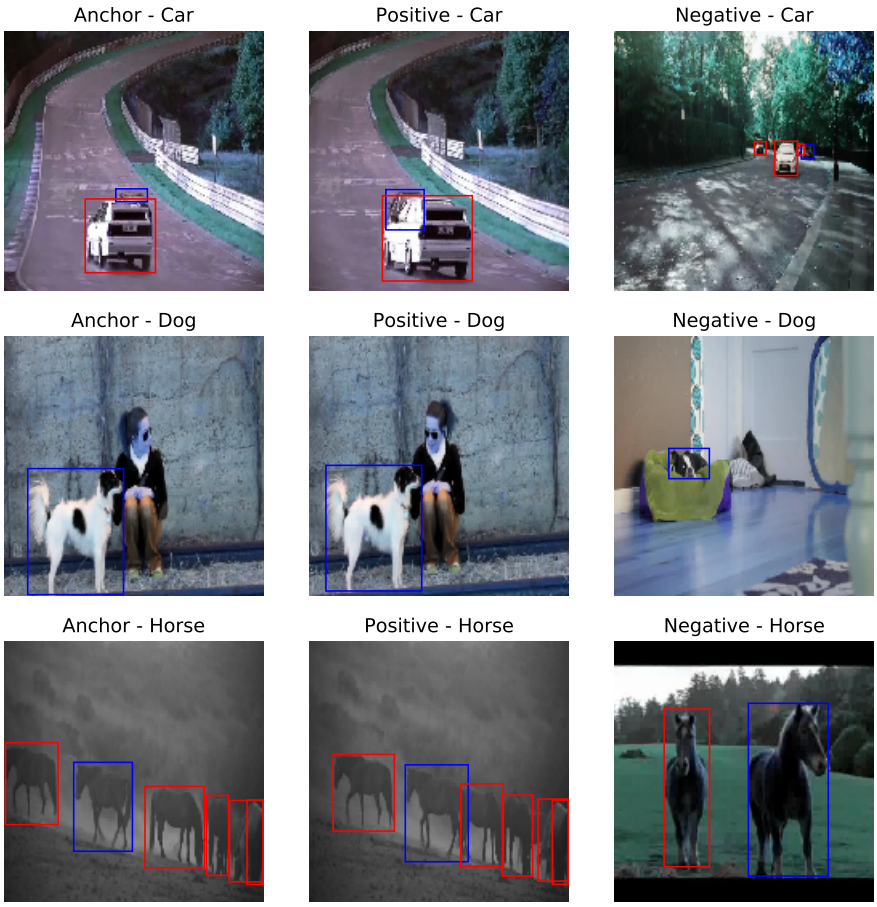


Figure 1: Samples of three triplets from the ImageNet Dataset: The blue coloured bounding boxes indicate the anchor, positive, or negative triplet. The red colored bounding boxes indicate the other objects in image. The negative sample have a different unique ID but the anchor and positive have the same unique ID.

jects from 2541 images for validation. The reduced dataset of objects is only 1% of the entire dataset but is at the required scale given the available resources for training. The multi-object images can contain objects that are not a candidate triplet, but we use all the objects in the selected images for detection and classification parts of our task. The said inclusion is necessary because our model is designed for multi-object predictions. The addition brings our training dataset to 47532 objects for training and 9132 objects for validation including the two variations per object. Since the images are in time-series we do see unique objects making appearances in different images. In that case, we use the additional variations for localization and classification but only the two designated instances are used as triplet members. Some of the samples of the dataset are given in Figure 1. All the three negative pairs in the figure are hard samples meaning they belong to the same category but are not the same object as the anchor and the positive object. This sample of the dataset also shows some of the challenges faced in detection, for example, the small scale of the car, and the extreme similarity of different horses.

4. Methodology

In order to create a model that is able to localize embeddings of objects in image space, we perform three tasks: classification, object localization (detection), and embedding generation. Classification acts as a measure of quality for object localization, and localization projects embeddings in a region occupied by an object. Towards that effort, we extend an existing model called "You Only Look Once" (YOLOB) [18] which can only localize and classify objects with a single pass-through of the model. We use the same ideology of a single model for multiple tasks to localize our embeddings. We filter the generated embeddings predicted by our model by bounding box filtering [18] based on conditional class probabilities and object probabilities.

The reduced dataset from Section 3 is used to mine triplets (anchor, positive, and negative) where every unique object is tagged as an anchor object/sample per epoch. For every anchor, there is a positive sample which has the same tracking id as the anchor object. The anchor sample and the positive sample are the same car pictured at a different time as seen in Figure 1. In the case a positive object is not available in the dataset, an augmentation of the anchor object is tagged as the positive sample. Augmentations as a backup is essential when scaling up the dataset as the frequency distribution of categories of images is not uniform. For the negative object, we first look for hard negatives meaning objects that are in the same category but are a different object compared to the anchor or positive object. If hard negatives are not available, objects of a different category are tagged as negative samples. Since the input to our model is an entire image with multiple objects, we encode the bounding box values and classes for all triplet and non-triplet objects, including the triplet tags into a target variable encoder used to compute loss for the model.

$$S * S(E + C + B(O, X, Y, W, H)) \tag{1}$$

The base architecture used for LocEm as shown in the Figure 2, is the ResNet50 architecture with pre-trained ImageNet weights. We found that our model benefits from the pre-trained weights as the video dataset from Section 3 has the same object categories as ImageNet Detection or Classification datasets. The base architecture is followed by a GeM layer which pools features from the Bottleneck layer of the ResNet50. The LocEm layers consist of a fully-connected layer with a LeakyReLu non-linear activation function and DropOut for regularization of the model. We modify the final layer of YOLOB by adding our embedding predictors of E units, as shown in Equation 1. The input given to the model is an image of size 448x448, and the model predicts bounding boxes B for every grid-cell of size S. Class probability C for classes in our dataset, object probability O, bounding box center coordinates (X, Y), and bounding box width and height (W, H) are predicted for every cell. Since the embeddings of objects are predicted per grid-cell, it enables our model to predict embeddings for every object in an image, and at their precise location in the image.

$$L_{le} = L_{loc} + L_{cla} + L_{conf} + L_{trp} \tag{2}$$

The loss function of LocEm (L_{le}) in Equation 2 is the summation of the localization loss (L_{loc}), classification loss (L_{cla}), box confidence loss (L_{conf}), and the triplet loss (L_{trp}).

$$\begin{aligned} L_{trp} = \lambda_{trp} \sum_{i=0}^{S^2} \mathbb{1}_{i,\gamma}^{obj} \Big\{ \\ max\{0, D(A,P) + G - D(A,N)\} \\ + max\{0, D(A,P) + G - D(A,N*)\} \Big\} \end{aligned} \tag{3}$$

The triplet loss in Equation 3 is calculated using the L2 distance denoted by function D, a margin factor G, and the localized embeddings of anchor A, positive P and negative N objects. Embeddings are predicted for every cell in the $S * S$ grid and we use the non-object (background) embeddings as additional negative embeddings $N*$ to calculate the triplet loss. The N sample provides a global negative sample relative to the dataset while $N*$ provides a local negative sample relative to the image space containing the anchor sample. The $\mathbb{1}$ denotes the presence of a triplet where γ (triplet flag) indicates the type of the triplet and λ_{trp} denotes the penalty to the loss. The other losses in Equation 2 from YOLOB are calculated using sum squared error and described in Section 8. The triplet loss and the classification loss are weighted equally. The target variable for input images encodes the localized ground-truth information of the objects as shown in the Equation 1. Though in the target variable, we replace E units with $\gamma = 1$, which acts like a flag to identify the triplet sample masks of objects for calculating the value of the triplet loss function during the model training phase.

The embeddings predicted for objects during the evaluation of the model are stored in a database. We use FAISS [8], a library for efficient similarity search as the database to store the predicted embeddings. After the model has been trained, it can predict the bounding box, the class, and the embeddings for multiple objects in an image. To store the embeddings in a database, they need to be uniquely identified for retrieval and fine-grained categorization. The identification is achieved by calculating the Intersection Over Union (IoU) of the predicted boxes and the target boxes. If the IoU is above a threshold, we can assign an id to the embedding of the predicted box. Since the composite key

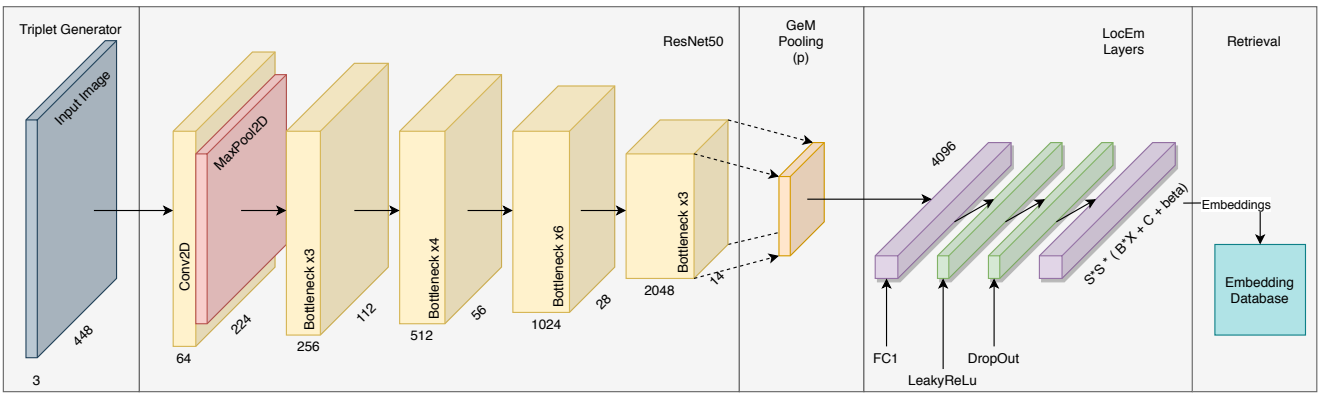


Figure 2: LocEm Model

identifies unique objects in the dataset, we use the composite keys as ids for the embeddings from the predicted boxes. The FAISS database is populated with embeddings from the training phase and separately for the validation phase. In either phase, after pouplating the datasbase, we use every embedding as a query embedding to be matched with another embedding, of the same unique object identifier (ID). A match is designated as successful if the ID of the query embedding is exactly same as the ID of the matched embedding. Only the Top-1 and Top-5 results with the least L2 distance from the embedding are considered for a successful match. The performance of the embedding generation of the model is revealed by the validation phase as the model never trains on the validation dataset. FAISS also provides methods to query the embedding dataset and retrieve the Top-1 and Top-5 matches based on a distance functions such as L2 distance. We use those methods for the fine-grained identification of predicted objects.

5. Experiments

We gauge the performance of our model by comparing its class prediction and identification retrieval with those of the other models in the literature. Since, to the best of our knowledge, our model is the only one that is able to perform detection, classification, and embedding generation in a single forward pass of a network, the comparisons were limited to models that performed a combination of two of the above tasks. Therefore, to create a pipeline for localized embeddings generation system, we use the other models but provide manual assistance for the missing tasks. This leads to an unfair comparison for our model, but our model's benefit is that it does not require any manual assistance as it can perform the three tasks to generate localized embeddings. Our goal for conducting these experiments is to understand the performance of our model, gauge its benefits and limitations, to ultimately chart a course for a future of localized embeddings in multi-object images.

For comparison, we use a ResNet50 only architecture (RSN50), the base Yolo architecture (YOLOB), and Multigrain architecture (MLTGR) [2]. All the models are trained on the same dataset and have the same augmentations. The augmentations include flipping, scaling, blurring, and shifting of objects along with random brightness, hue, and saturation variations. Stochastic Gradient Descent (SGD) is used for optimization with momentum=0.9 and weight decay=$1e^{-4}$. A variable learning rate schedule is used except for RSN50 and MLTGR which have a predetermined learning rate that also performed better in our tests.

The model RSN50 is used as a baseline for class prediction. The final layer of ResNet50 is replaced with a softmax layer of units equal to the classes in our dataset. The final feature map before the softmax layer is used as the embedding representation for the objects in our images. The feature map is flattened leading to a vector of size 2048, which is used as the number of dimensions for the predicted embeddings. Since the model is not trained for detection, we crop the objects from the images before they are used as inputs to the model. By cropping the objects, we can map the embedding to the object and assign it to the unique identification for retrieval and fine-grained categorization.

The MLTGR model is used as a baseline for embedding generation. It uses a combination of cross-entropy loss and margin loss to predict embeddings for the entire image containing only one object. The embeddings are generated by applying L2 normalization over the classification layer consisting of 30 units for the categories in our dataset. The default embedding size of 2048 units is used for our experiments. The model enables classification and requires cropped images as inputs because it does not support localization. We make a few changes to the original model that was designed to be trained on the entire ImageNet dataset. We modify the input image size from 224x224 to 448x448 and use the same augmentations as we used for other models. We changed the augmentation list and the image resolution for a fair comparison. Our experiment utilizes the

Repeated Augmentation Sampler (RAS) provided with the model. RAS uses augmentations of images to generate positive samples required for its Margin Loss. The positive and negative pairs are generated from the images in a batch used for training. The model does not require the variation in object poses or movements provided by our dataset, but they are provided to compare the dataset's performance with other models. We set the number of repeated augmentations to 2 but reduce the recommended batch size to 64 to adhere to resource constraints invoked from using a higher resolution of images. The pooling exponent required for the model's generalized mean pooling layer is set to the 3 as it yields the best performance [2].

The model YOLOB is used as a baseline for object detection, which is refined based on the class probabilities predicted by the same model. The final layer of the model predicts class probabilities for every grid on the image, and we only modify the number of classes to be predicted for our dataset. The implementation used for YOLOB[1] was a port of the darknet framework for Pytorch though we use the ResNet50 architecture for a fair comparison. The implementation modifies the final layer of YOLOB by replacing its linear activation with sigmoid activation. We found that this replacement was necessary as, without it, the pretrained ResNet50 predicts negative integers for the width and height of the bounding boxes. To extract the embeddings of objects, we need to make a second forward pass of the model. The second pass is required since the last feature map of its CNN encodes information about the entire image with multiple objects, and therefore it would be improbable to extract embeddings for individual objects. The objects could have been been cropped before training the model, and its last feature map could be used as its embedding in a single pass through. Though, if the model is trained on cropped images with single objects, it defeats the multi-object prediction design of YOLOB and the goal of our comparison task. In the second passthrough we provide the model with cropped out object so we can compare its emebdding generation without an impact from its own localization performance.

To overcome the limitations of the models mentioned above, we train our LocEm model that can learn the three tasks simultaneously. With the same base architecture of ResNet50, we construct the final layer as described in Equation 1. The number of bounding boxes is set to B=2, predicted for every grid-cell in a grid of size S=7. The number of classes from our dataset is C=30, and the embedding predictors are set to E=64. By projecting the embeddings into the units of the final layer, we can vary its dimensionality before training the model unlike embeddings extracted from the CNN feature maps of other models.

Our model is trained on 64 batches consisting of 192 images yielded by the generator. We gauge the training time performance of the model by constructing a measure of conditional classification accuracy in addition to the individual losses for the three tasks. Classification accuracy measurements are conditioned on the object belonging to a grid-cell that contains the center of the object. We partially decode the predicted tensor based on the location of objects from the target tensor to check the classification accuracy during validation. The object probability O and triplet flag γ unit in the target tensor are used as flags to identify the objects. This method of partially decoding the predicted tensor is used for validation only while training and to avoid the cost of decoding the entire predicted tensor.

In the model evaluation phase, we decode the entire predicted tensor. The implementation of LocEm provides a modified YOLOB detector to detect the object bounding boxes, its probabilities, and classes for validation. The detector decodes the predicted tensor and applies non-maximal suppression to filter the bounding boxes. In the detector, we also extract the predicted embeddings based on the object confidence conditioned on its class probability. This modification ensures that the extracted embeddings are localized and belong to an object with a certain confidence. We use unique object ID (UID) to identify identical objects and class ID (CID) to identify objects that belong to identical classes. The UID matches reveal the fine-grained categorization performance of the model whereas the CID matches reveal the coarse-grained categorization performance of the same model. To validate the model, we implement the method described in Section 4 to assign the UID or the CID to the correct object. The said method is also used to calculate the Top-1 and Top-5 class accuracies of the objects. For retrieval, we separately add the embeddings to the embedder database for the training and validation phases of LocEm. The embeddings from the respective phase are used as query embeddings on the database of the same phase. Every query embedding is required to find a correct match out of the remaining dataset which is 47532 objects for the training phase and 9132 for the validation phase. Based on the top retrieved indices post-query, the Top-1 and Top-5 fine-grained categorization accuracy for both the phases are calculated separately.

6. Results

This section provides the performance results of the models discussed in Section 5 and their analysis. The following results in this section are the performance of the validation dataset post-training the model and the post-processing of the decoded tensor. The post-processing method is part of the single passthrough model implementation. To gauge the performance of our model, we use the mean average precision (mAP) metric that can accurately represent the performance of localization and clas-

[1] https://github.com/motokimura/yolo_v1_pytorch

sification. The average precision is calculated with an Intersection over Union threshold of 0.5, as is standard. We separate the classification results as directly predicted by the model and by a secondary method using only the embeddings to represent the class of an object. We employ the secondary method only to judge the representability of our embeddings on a task other than unique object retrieval for fine-grained categorization.

We vary the length of an embedding to find a length that minimizes the triplet loss on a reduced dataset size. We found that embeddings of length 64 were an optimal length for our task and increasing the length up to 2048 led to memory constraints. The memory constraints stem from the factor that embeddings are predicted for every cell, increasing the number of units in the final layer. The same increase also leads to the model having to learn more values to represent an object, thereby degrading the quality of the model. Other factors that led to a memory constraint were increasing the size of the dataset from 2 to 4 samples per unique object and architectures with more parameters such as ResNet101. We also found that the performance improvements with the GeM layer and using background embeddings as negative embeddings realized gains to warrant their inclusion without any significant additional computation. We arrived to the same conclusion that GeM pooling outperforms [26] average pooling for encoding features. The above-mentioned parameters of the network including the size of the fully-connected networks and the loss function penalties were tuned using a separate slice of the training dataset. The validation dataset is only used for the evaluation phase.

6.1. Classification and Localization Results

Model	Top-1 (%)	Top-5 (%)	mAP(0-1)
RSN50	77	**93**	-
MLTGR	74	91	-
YOLOB	71	81	0.20
LocEm	**78**	88	**0.22**

Table 1: Validation Classification and Localization Results

In the first set of experiments in Table 1, we compare each model's classification capability. The mAP results display the classification performance with localization performance factored in, whereas the Top-1 and Top-5 results display classification without a penalty from the localization performance. The distinct metrics are necessary to isolate the classification performance of models that do not perform localization.

As shown in Table 1, we see that the RSN50 model has the better classification performance in the Top-5 task with a difference of 5%, but LocEm does equivalently well on the Top-1 task. The single percentage point difference in the Top-1 task is insignificant due to the rounding up of the final results. The classification results were expected as such since the RSN50 model is trained to perform a single classification task compared to the multi-task training of the other models. Furthermore, the RSN50 model does not localize the objects, and therefore it is provided with objects manually cropped from multi-object images. Both these benefits avoid the errors in localization present in the YOLOB and LocEm models. The MLTGR model performs similarly except for a 3% drop in the Top-1 and 2% drop Top-5 task compared to RSN50. The MLTGR model also uses manually cropped object images, but it can perform two tasks simultaneously, i.e., classification and embedding generation, while avoiding any localization errors. The RSN50 and MLTGR models serve to provide a baseline of classification and embedding generation, respectively. They are, in effect, used in a localized embedding generation system but with assistance in the crucial task of localization. The YOLOB model, on the other hand, suffers the same localization errors as LocEM. However, our model shows a marginal improvement in the mAP metric, and 7% improvement in the Top-1 and Top-5 task compared to YOLOB. This gain could be attributed to the triplet loss acting as an additional discriminator in the overall loss function of LocEm.

The purpose of these validation classification results is to show the impact of the multi-task losses of LocEm when comparing with the other models. LocEm's localization and embedding generation losses do not significantly affect its classification performance. Thus, its performance is comparable to a dedicated classification model such as RSN50 and a multi-task localization model such as YOLOB.

6.2. Fine-Grained Categorization

Model	Train T1	Train T5	Val T1	Val T5
RSN50	50	65	49	68
MLTGR	56	72	59	**76**
YOLOB	32	41	34	48
LocEm	**69**	**82**	**61**	75

Table 2: Unique ID Retrieval Results (expressed as percentages) for fine-grained categorization: T1 and T5 denote the success of a match in the Top-1 and Top-5 retrieval results

In the second set of experiments in Table 2, we compare each model's unique object retrieval capability for fine-grained categorization. The results in the table are separated by the training or validation phase of the dataset. In each phase, all the results factor in the location of the object embeddings.

Localization of objects is crucial to embedding generation as the representability of an object is also conditioned

on its location in the image space. We see that LocEm performs better than the other models through both the training and validation phase except for comparable performance in Val T5 task. All the other models have access to manually cropped-out objects and do not perform localization. We observed that LocEm detects 93% and 70% of all the unique objects in our experiments' training and validation phase, respectively. This observation highlights the trade-off between models that can perform localization as opposed to the models that require manually cropped objects. As a result we do not claim that LocEm outperforms the other models. In the validation phase, uniquely identifying objects is more challenging since the model has not learned from the validation images. The embedding generation relies on the model to have learned to differentiate between unique objects that it has never seen before. The task is even more challenging when different objects are extremely similar such as the horses seen in Figure 1 in Section 3.

The model YOLOB has the most performance difference compared to our model because its extracted predicted embeddings do not represent just the object but other information that it predicts, such as bounding box and class probabilities. The results of YOLOB highlights the importance of localized embeddings in multi-task models such as MLTGR and LocEm. MLTGR performs better than RSN50 and YOLOB because it is dedicated to learn embeddings via its loss function. From these experiments, we can deduce that models that include dedicated embedding generation using loss functions such as MLTGR's margin loss and LocEm's triplet loss have a considerable positive impact on embedding generation.

6.3. Coarse-Grained Categorization

Model	Train T1	Train T5	Val T1	Val T5
RSN50	**98**	**99**	**94**	**98**
MLTGR	90	96	**94**	**98**
YOLOB	58	78	66	84
LocEm	86	92	84	91

Table 3: Class ID Retrieval Results (expressed as percentages) for coarse-grained Categorization: T1 and T5 denote the success matches in the Top-1 and Top-5 retrieval results

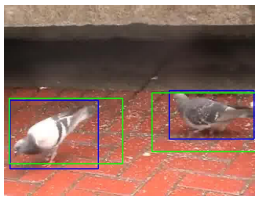
In the third set of experiments in Table 3, we test the expressibility of the embeddings for a coarser task such as classification. The results are separated into the training and validation phases while also factoring in the location of objects. The said task is coarser as a query object can be matched to multiple other objects in the same category and need not be the same as the query object. This task increases the potential number of matches considerably since more object samples exist for an entire category than for a unique object ID. The increase in matches explains the performance difference between the classical classification task in Table 1 compared to matching embeddings. The larger search space enables a more accurate classification, albeit at the cost of a database search.

From the Table 3 results, it is evident that LocEm is unable to beat the embeddings from the RSN50 and MLTGR models. Though, it must be highlighted that the LocEm embeddings were generated for a different task, fine-grained categorization. The embeddings of LocEm are learned to identify unique objects, whereas the embeddings of MLTGR are dedicated to classification performance. The embeddings extracted from RSN50 are also from a dedicated classification model. Just as in the previous experiments, only LocEm performs localization while other models are assisted with cropped-out objects. In the case of LocEm, the defined triplet pairing system is designed towards matching unique objects together and separating them from other objects that belong to the same category. The said pairing system is beneficial for identifying unique objects but leads to a distance in embeddings that belong to the same class. In our model, the sifting of image objects into the three categories of anchor, positive, and negative allows the ability to define a similar pair and a dissimilar pair for any given application. If an embedding-based classification is the objective, then a similar pair in a triplet can be assigned to different objects from the same class. Matching embeddings is beneficial when softmax does not perform optimally for classes greater than 1000.

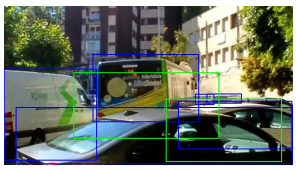
This experiment aims to test the representability of embeddings, primarily when they are generated for a different task but tested on a new task. The embeddings generated by LocEm can be generalized for a coarser task, without any retraining, but with a performance trade-off.

6.4. Model Introspection

We introspect our model in order to understand the performance of our model. Even though all our images of objects come from the same database, they have a considerable variation in certain aspects of the objects in a given image. Since these are natural images, factors such as the variability in object location and resolutions can play a significant role in the predictions on different images. As seen in Figure 3 (a), the model performs very well when there are objects with some separation between them. But, there are also instances such as Figure 3 (b) where multiple objects are cluttered extremely close or are occluded. In the latter cases, bounding boxes can overlap, leading to misidentification. We have also seen images where occlusions as in Figure 3 (c) can cause the detection to fold into a single bounding box. The bicycle object in Figure 3 (d) looks blended in because of the scene lighting leading to a similar color palette for the bicycle and the tree next to it. These are similar challenges faced by the YOLOB model that also

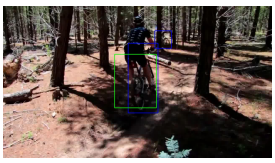


(a) Objects with a seperation



(b) Multiple objects in close proximity

(c) Occluded objects



(d) Objects blending with background

Figure 3: Variation of predicted images from LocEm. The green coloured boxes are the predicted boxes and blue coloured boxes are the ground-truth boxes.

struggles with occlusions and objects in close proximity.

7. Conclusion

We proposed the design for a single pass-through multi-task model named LocEm that can detect objects, classify, and localize embeddings of multiple objects in an image through the previous sections. Our model design includes modifying the triplet loss function to encode background embeddings as an additional measure of local negative samples for objects. We defined a method to repurpose an existing large-scale dataset, ImageNet Videos, to generate triplets with natural augmentations such as pose and action movement variation. In order to compare our multi-task model with other models in the literature, we proposed modifications to the other models to extract object embeddings. Even though our model does not always beat the performance of the other models, it must be highlighted that our model always performs a task more than the others. Localization is crucial to the task of classification and embedding generation. While the other models are assisted with manual localization, the localization performance of our model plays a significant role in the comparative results. The benefit of LocEm lies within its unique ability to perform the three tasks in a single pass-through, on multi-object images, and without the need for any manual assistance. We do see a scope for improvements in various aspects of the model to help it reach its potential.

There are a few avenues of potential improvements that could be made to our model. More samples of objects can be included per object with additional hardware resources so that the model has a larger image space to learn the positive pairs. As a result, we can capture a more significant variation in object movement and pose with more positive pairs. Since a positive object can also be used as a negative object for another triplet, the scaled dataset could translate to better and harder negative pairs to distinguish between objects that look very similar. Multi-task models provide the benefit of learning multiple tasks and balancing the weight distribution of their respective loss function can lead to significant improvements. The cost of balancing the weight distribution in terms of resources would not outweigh its benefits. An improvement to the localization performance will directly influence the model embedding generation performance. Standardized box variations could be used as a starting point in images to act as better heuristics to predict boxes than initial weights from the model. Some of the performance of our model can be directly attributed to the dataset. Addressing occlusion, objects in close proximity, differentiating extremely similar objects remain an open challenge in deep learning-based computer vision tasks.

The benefits of localizing embeddings are not limited to the image domain. Even though we use images as an application for our model, embeddings are generated for other data types such as sentences and graphs. Our future work will explore the ability to use localized embedding generations in graphs to identify sub-graphs in applications and generate its embeddings. One of the challenges in the graph domain includes localization of sub-graphs in non-euclidean space. The said application includes codes expressed as control flow graphs that can be used to identify sub-graphs and match with optimized versions of those codes for execution performance improvement.

8. Related Equations

$$
\begin{aligned}
L_{loc} &= L_{xy\_loss} + L_{wh\_loss} \\
L_{xy\_loss} &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 \\
&\quad + (y_i - \hat{y}_i)^2)] \\
L_{wh\_loss} &= \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 \\
&\quad + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]
\end{aligned}
\tag{4}
$$

The LocEm loss funtion L_{le} in Equation 2 is the summation of the location loss L_{loc}, classification loss L_{cla}, object confidence loss L_{conf}, and the triplet loss L_{trp}. The first three losses are devised in YOLOB with their explanation in this section. The explanation for our triplet loss is provided in Section 4. The summations in Equations 4,5 and 6

are for all the B bounding boxes calculated per grid location i in an SxS grid. The presence of an object for bounding box j in grid i is indicated by a boolean $\mathbb{1}_{i,j}^{obj}$.

The location loss in Equation 4 is the summation of the bottom-left corner (x, y) of a bounding box, and the w width and h height of the box. It's calculated using sum squared error. The square root of the width and height are taken to address the difference in loss values of the variance in large and small bounding boxes.

$$L_{cla} = \lambda_{cla} \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (5)$$

In Equation 5, the classification loss is calculated for every cell where p_i is the probability of the class c conditioned on the presence of an object in a given box. At test time the conditional probability is multiplied with object confidence, and the IoU score of the predicted and ground-truth box. A max over the classes is calculated in the preprocessing step following model inference.

$$\begin{aligned} L_{conf} = &\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\ &+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \end{aligned} \quad (6)$$

The object confidence loss in Equation 6 is also a summation of confidence of boxes with and without an object. To avoid the gradients being overpowered from bounding boxes with no objects a lower penalty of λ_{noobj} is included and a higher penalty λ_{coord} is included for the bounding box location loss.

Acknowledgements

I would like to thank the following people involved in the conception of this work. It includes, Prof. Humphrey Shi, Prof. Boyana Norris, Prof. Daniel Lowd, Prof. Dejing Dou, and Prof. Joe Sventek for their feedback, discussions, suggestions, and resources. In addition, the ideas presented in this paper stem from the many conversations with my various colleagues in the Department of Computer and Information Science, University of Oregon.

References

[1] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *European conference on computer vision*, pages 584–599. Springer, 2014.

[2] M. Berman, H. Jégou, A. Vedaldi, I. Kokkinos, and M. Douze. Multigrain: a unified image embedding for classes and instances. *arXiv preprint arXiv:1902.05509*, 2019.

[3] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[4] A. Gordo, J. Almazán, J. Revaud, and D. Larlus. Deep image retrieval: Learning global representations for image search. In *European conference on computer vision*, pages 241–257. Springer, 2016.

[5] H. Jégou and O. Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *European conference on computer vision*, pages 774–787. Springer, 2012.

[6] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *International journal of computer vision*, 87(3):316–336, 2010.

[7] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3304–3311. IEEE, 2010.

[8] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[10] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[11] A. Mikulik, M. Perdoch, O. Chum, and J. Matas. Learning vocabularies over a fine quantization. *International journal of computer vision*, 103(1):163–175, 2013.

[12] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2161–2168. Ieee, 2006.

[13] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[14] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3384–3391. IEEE, 2010.

[15] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2007.

[16] F. Radenović, H. Jégou, and O. Chum. Multiple measurements and joint dimensionality reduction for large scale image search with short vectors. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, pages 587–590, 2015.

[17] F. Radenović, G. Tolias, and O. Chum. Fine-tuning cnn image retrieval with no human annotation. *IEEE transactions on pattern analysis and machine intelligence*, 41(7):1655–1668, 2018.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[19] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[20] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[21] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *null*, page 1470. IEEE, 2003.

[22] G. Tolias, Y. Avrithis, and H. Jégou. Image search with selective match kernels: aggregation across single and multiple images. *International Journal of Computer Vision*, 116(3):247–261, 2016.

[23] G. Tolias and H. Jégou. Visual query expansion with or without geometry: refining local descriptors by feature aggregation. *Pattern recognition*, 47(10):3466–3476, 2014.

[24] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of cnn activations. *arXiv preprint arXiv:1511.05879*, 2015.

[25] P. Turcot and D. G. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problems. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 2109–2116. IEEE, 2009.

[26] L. Zheng, H. Zhang, S. Sun, M. Chandraker, Y. Yang, and Q. Tian. Person re-identification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1367–1376, 2017.