# Deep Visual-Semantic Embedding Models for Mobile

**Swarna Saraf**
Department of Computer Science
Stanford University
`swsaraf@stanford.edu`

## Abstract

This project uses deeplearning methodologies to implement a deep visual-semantic embedding model for mobile devices. Such models enable usage of novel image queries in many mobile applications to support features such as image tagging and retrieval. These models also help offset the problems associated with acquiring sufficient labeled data samples for the ever exploding image categories, by leveraging semantic information from word embeddings such as fastText [12] to make predictions about the labels not observed during training. A lightweight mobile architecture SqueezeNet 1.1 [4] is used to train a model with pretrained fastText word vectors to learn semantic relationships between image labels and map images into a rich semantic embedding space. This project observes that such visual-semantic models are able to perform image-to-image, image-to-text and text-to-image associations with reasonable accuracy while using less than 7% of disk space and train parameters as compared to Resnet34.

## 1 Introduction

Mobile devices, with ever increasing on-device memory and better camera resolutions are the most popular means of clicking and storing pictures. For running deeplearning models on mobile devices, models should have smaller memory footprint and should give reasonable performance. A deeplearning visual model that is able to leverage semantic information from word representation systems such as fastText [12] or GloVe, will not see image labels as disconnected (as traditional convolutional neural network architectures do) and will be able to transfer semantic information from learned image labels to previously unseen or new labels.

The project implements a deep visual-semantic model based on Squeezenet 1.1 [4] architecture and trained with fastText [12] word vectors. The model takes in image input and provides a 300-D image feature vector output. Using efficient cross-platform similarity search library such as nmslib, the output feature vector can be used for image similarity search in model predictions, or for label prediction based on lookup of the nearest fastText [12] word vector representation for the known image labels in dataset. The model can also be generalized for zero-shot use cases by performing nearest neighbor search in model predictions for the fastText [12] word vector representation of the input text label.

## 2 Related work

### 2.1 Mobile CNN Architectures

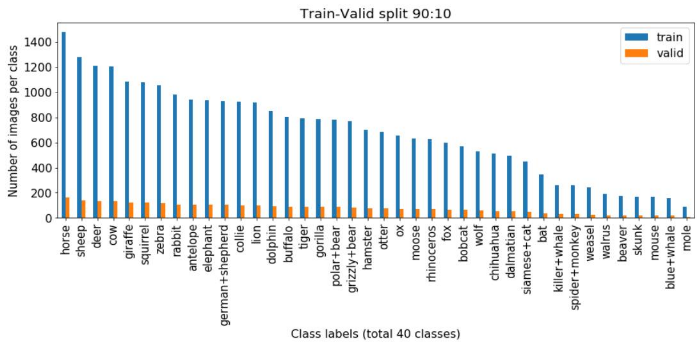
Squeezenet 1.1 architecture [4] (implementation in [11] ) employs novel techniques such as use of fire modules to reduce model parameters while maintaining reasonable accuracy. This project preserves the performant fire modules (convolutional layers) while discarding the softmax classifier head. A custom head is introduced with a softmax layer (along with average/maxpooling layers and a bunch of linear layers) with outputs equal to number of train classes in the AWA2 [8] dataset.

### 2.2 Deep Visual-semantic model approaches

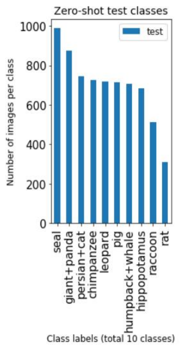
This project is influenced by the work in Frome et.al.[2] with some differences. Instead of training a language model from scratch, this project uses 1 million pretrained word vectors 300-D obtained by training fastText [3] on Wikipedia 2017, UMBC webbase corpus and statmt.org news dataset (16B tokens). This is a much bigger embedding space than the one used in [2]. Not having to train a language model also reduces the number of training phases to two instead of three. This project does implement a visual model from scratch. The choice of visual model architecture is what makes our project unique and interesting. [2] uses ILSVRC 2012 winning model architecture while this project uses a very lightweight model architecture Squeezenet 1.1 [4] suitable for deploying on mobile and embedded devices. Popular Resnet34 model architecture is used as a baseline for both the training stages. The visual model used in this project is pretrained on ImageNet. While [2] uses ImageNet dataset, this project uses AWA2 dataset proposed in [6] as the benchmark dataset for zero-shot learning methods. Because of the above mentioned differences in implementation, we cannot do a 1:1 comparison of the results or draw a direct conclusion.

### 2.3 Zero-shot learning methods and techniques

The project uses AWA2 [8] dataset proposed in [6], for performing zero-shot learning tests on the 10 class hold-out set. The project requires significantly more work for getting reasonable Top-1 accuracy and Top-3 accuracy with lean models geared towards mobile devices. The project does measure per class averaged top-1 accuracy (i.e. prediction is correct when the class predicted is correct) on validation set as the dataset is not well-balanced with respect to the number of images per class.



(a) Train-valid split 90:10



(b) Zero-shot test classes

Figure 1: Image distribution in the dataset

## 3 Dataset and Features

The dataset used in this project is Animals with Attributes2 [8] (further discussed in [6]). It is a benchmark dataset for transfer learning algorithms, such as zero-shot learning. The dataset contains 37,322 images for 50 animals classes. On average, each class contains 746 images. There is a class imbalance where the least populated class, mole has 100 images and the most populated class, horse has 1645 images.

Image distribution for the AWA2 dataset can be seen in Figure 1. 30,337 images in the 40 train classes are split 90:10 to create train and valid sets. A test set containing 6985 images belonging to 10 separate classes are set aside purely for zero-shot tests. Data augmentation techniques are used to help models generalize better. Pixel and coordinate transforms such as flip, rotate, warp, zoom, lighting transforms are applied in an optimized way using fastai [9] library.

## 4 Methods

The project uses Squeezenet 1.1 [4] model architecture pretrained on ImageNet dataset. This model architecture is lean and suitable for mobile devices. The model creation and training is a two stage process, as shown in Figure 2.

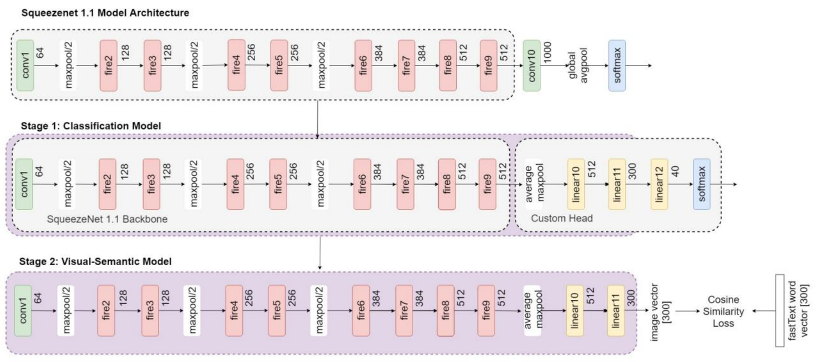


Figure 2: Architecture for Stage 1 and Stage 2

In Stage 1, a multi-class classification model is built using Squeezenet 1.1[4] model architecture backbone containing convolutional layers (and fire modules), rest is discarded. A custom head is initialized containing avg/max pooling and a few linear layers. *Transfer learning* is used to retrain only the layers in the custom head (while keeping the body weights frozen). Next, the layers in the body are unfrozen and the whole model is retrained using differential learning rates. This model is trained for accuracy and uses Cross Entropy loss for measuring performance.

In Stage 2, a visual-semantic model is built by using the saved model from Stage 1 and discarding the softmax layers so that the model outputs a 300-D image feature vector. The body weights are frozen and linear layers are retrained to minimize the cosine loss between image feature vectors and pretrained fastText word vectors. Once again the layers in the backbone are unfrozen and the entire model is retrained using differential learning rates. In this stage, the cosine similarity loss function is used. This is a non-metric loss function well suited for similarity comparisons in high dimensional spaces.

$$similarity = cos(x1, x2) = \frac{x1 \cdot x2}{||x1|| \cdot ||x2||} \tag{1}$$

## 5 Experiments/Results/Discussion

### 5.1 Hyperparameter Search and tuning

One cycle policy [1] is used to train the network faster. 1. Learning rate($\alpha$): Mock training is done with varying $\alpha$ and losses are determined. Then an optimum value of $\alpha$ is chosen where loss still improves 2. Number of frozen layers (lf): At each stage, lf is varied and model is fine tuned 3.Momentum:(0.85, 0.95) 4. Weight decay:0.01, 5.Optimization:Adam ($\beta$1=0.9,$\beta$2=0.99) Dropout regularization is used along with a final batch normalization layer to train the model.

### 5.2 Model comparison with Baseline

Resnet34 model is used for baseline comparison. The SqueezeNet based model performs slightly worse in the Classification task, while using less that 7% disk space and trainable parameters.

| **Stage 1 Model** | **Total Parameters** | **Size on Disk(KB)** | **Epochs** | **Accuracy** |
|---|---|---|---|---|
| Resnet34 | 21979164 (100%) | 85980 (100%) | 8 | 95.26% |
| Squeezenet 1.1 | 1416988 (6.45%) | 5565 (6.47%) | 8 | 88.66% |

Table 1: Stage1 comparison with baseline

| Stage 2 Model | Total Parameters | Size on Disk(KB) | Top1 Accuracy | Top3 Accuracy | Top5 Accuracy | Top10 Accuracy |
|---|---|---|---|---|---|---|
| Resnet34 | 21967044 (100%) | 85932 (100%) | 61.79% | 77.68% | 83.19% | 89.19% |
| Squeezenet 1.1 | 1404868 (6.40%) | 5517 (6.42%) | 80.26% | 90.25% | 92.74% | 96.05% |

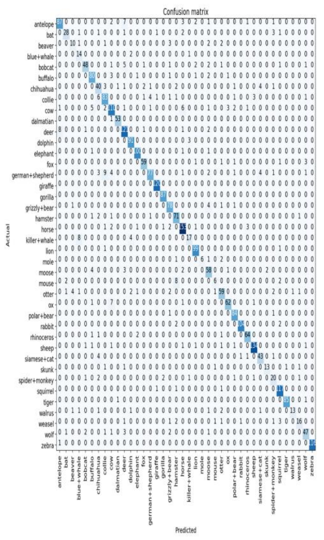
Table 2: Stage2 comparison with baseline (8 epochs each)

| Zero-Shot Results (Stage 2) | Top5 Accuracy | Top10 Accuracy |
|---|---|---|
| Resnet34 | 26.53% | 42.29% |
| Squeezenet 1.1 | 36.85% | 52.03% |

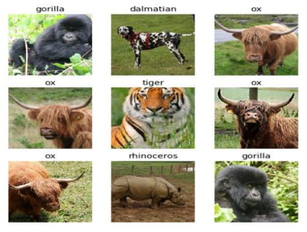
Table 3: Stage2 Zero-Shot results

## 5.3 Stage1: Confusion Matrix/Error Analysis

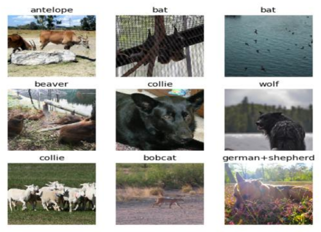
In Stage 1, the CNN model is trained for accuracy for multi-classification. Given the class imbalance, Confusion matrix seems a good choice for interpreting model results. In Figure 3(b) and 3(c), images in top losses are shown along with their heatmaps. In Figure 3(d) and 3(e), images in smallest losses are shown along with their heatmaps. The heatmap analysis indicates similar areas are activated for a particular class, probably indicating certain salient features for the class. For example, for an ox areas around nostrils and horns are activated. Also, images with animal close-ups seem to have minimum losses as their features are more visible.



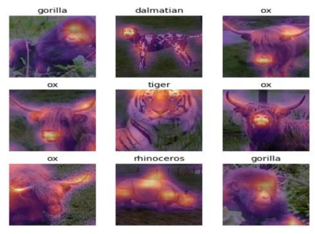
(a) Confusion Matrix



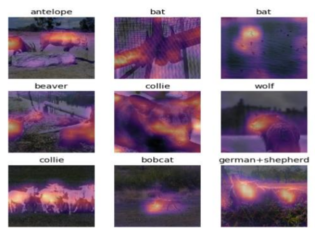
(b) Min loss images



(d) Max loss images



(c) Min loss images' heatmaps



(e) Max loss images' heatmaps

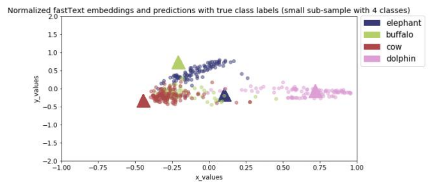
Figure 3: Stage 1 Results and Error Analysis

## 5.4 Stage2: PCA Analysis

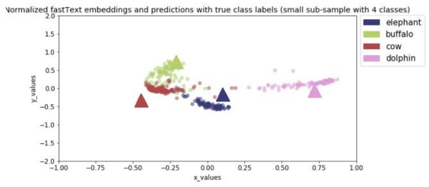
PCA Analysis is done in Stage 2 to visualize the image feature vector representation for a sub-sample of the image classes in 2D. Four animal categories are chosen and their feature vector representation for all valid samples are PCA decomposition are plotted after epoch-1 and epoch-8 as shown in Figure 4.

## 5.5 Results

Squeezenet 1.1[4] based visual-semantic model gives per class-averaged Top-1 accuracy as 66.88%. Some other results are demonstrated below:



(a) After Epoch 1

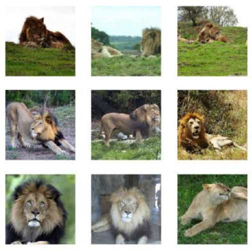


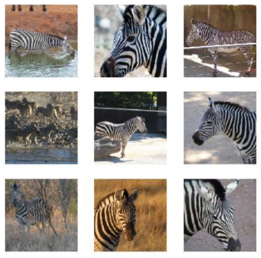
(b) After Epoch 8

Figure 4: PCA Analysis of normalized fastText embeddings and model predictions. Triangles represent true fastText representations for given labels.

### 5.5.1 Text to Image

Figure 5 shows k Nearest Neighbor search in model predictions using fastText embedding for provided text.


(a) Text query: 'king'


(b) Text query: 'stripes'

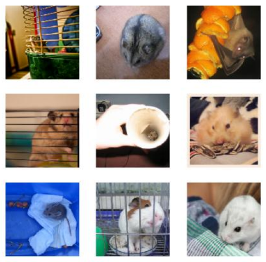
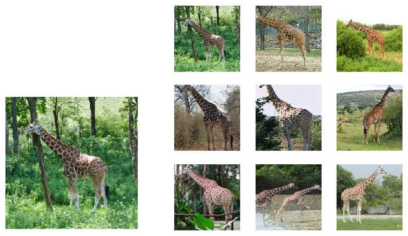

(c) Text query: 'computer'

Figure 5: Text based search in the model predictions.

### 5.5.2 Image to Image

Figure 6 shows k Nearest Neighbor search in model predictions using model output for a given input image. Example from zero-shot test data set is shown in Figure 6(b)


(a) Image Search: giraffe image

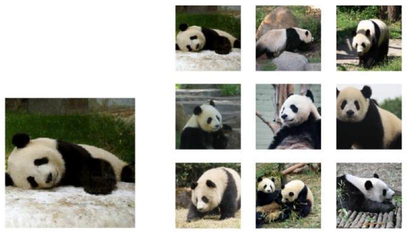

(b) Image Search: panda image

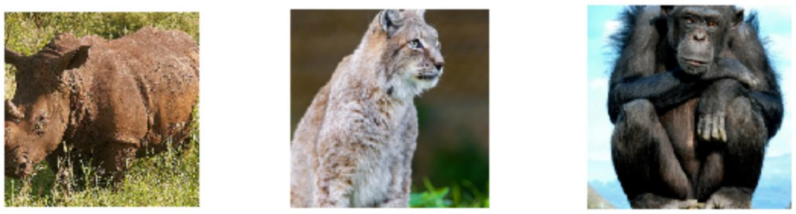
Figure 6: Image similarity search on model predictions

### 5.5.3 Image to Text

Figure 7 shows top-5 labels for an image based on nearest neighbor search for model prediction in fastText embeddings for all class labels. (c) demonstrates prediction on test set.

## 6 Conclusion/Future Work

This project demonstrates that it is feasible to build lightweight visual-semantic models for mobile applications while meeting acceptable performance threshold. Applications with features such as image search (based on similarity or complex text queries), tag generation, cataloging new products (zero-shot learning) can make use of such models.



(a) ['rhinoceros', 'elephant', 'deer', 'antelope', 'moose']

(b) ['dalmatian', 'bobcat', 'beaver', 'skunk', 'chihuahua']

(c) ['gorilla', 'chimpanzee', 'elephant', 'hippopotamus', 'zebra']

Figure 7: Class label prediction for image. True labels (a) rhinoceros (b) bobcat (c) chimpanzee

Future work includes (1) Improving zero-shot learning performance for Squeezenet 1.1 model. (2) Exploring techniques, aside data augmentation, to deal with class imbalance issue. (5) Using other popular word representation libraries such as GloVe to get the embedding vectors. (4) Extending the concept of semantic embeddings to augment audio datasets.

## 7 Contributions

This project has a single contributor. The project github repo is https://github.com/swarna04/cs230.

## References

[1] Leslie N Smith. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018.

[2] Andrea Frome, Gregory S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *NIPS*, pages 2121–2129, 2013.

[3] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext.zip: Compressing text classification models, 2016. cite arxiv:1612.03651Comment: Submitted to ICLR 2017.

[4] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360, 2016.

[5] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[6] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning-a comprehensive evaluation of the good, the bad and the ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[7] TorchVision Models. pytorch.org/docs/stable/torchvision/models.html.

[8] Animals with Attributtes 2 Dataset. https://cvml.ist.ac.at/awa2/.

[9] Fastai Deep Learning Library. https://github.com/fastai/fastai.

[10] Pytorch Library. https://pytorch.org/.

[11] Squeezenet Architecture. https://github.com/deepscale/squeezenet.

[12] FastText English Word Vectors. https://fasttext.cc/docs/en/english-vectors.html.

[13] Non-Metric Space Library (NMSLIB). https://github.com/nmslib/nmslib.