Deep Visual-Semantic Embedding Models for Mobile
Swarna Saraf (email: sswaraf@stanford.edu)

Motivation
This project implements a deep Visual Semantic Embedding model for mobile devices. Such models enable usage of novel image queries for image tagging and retrieval. These models also help offset the problems associated with insufficient labeled samples for the ever-expanding image categories.

A lightweight mobile architecture SqueezeNet 1.1 [2] is used to train a model to associate images classes with pre-trained fastText word vectors for the corresponding class labels. Semantic information from the word vectors (embeddings) augments the classification model for many interesting applications as demonstrated in this project.

![Visual Domain](Image)

Semantic Domain

Data
AWA2: Benchmark dataset for transfer learning algorithms, such as zero-shot learning. 37322 images of 50 animal classes, 13 GB in size.

Train/val split: 90/10 of 40 classes (30387 images)

Test set: 10 classes (6985 images) are kept aside for zero-shot learning tests.

Data Augmentation techniques help models generalize better.

Pseudo and coordinate transforms, such as flip, rotate, warp, zoom, lighting transforms are applied in an optimized way using Faiss library [4].

Pre-trained word vectors (300 dimensions) trained using fastText are used. 1 million word vectors trained on Wikipedia 2017, UMBR web-base corpus and stanford.org news dataset (5GB tokens).

Model
Stage I: A multi-class classification model using SqueezeNet 1.1 [2] model architecture backbone with a custom head (comprising of linear layers) is trained for accuracy.

Loss function: Cross entropy loss is used to measure model performance.

Stage II: A regression model is used to train 300d image feature vectors (obtained by discarding the softmax layer) with pre-trained fastText word vectors.

Loss function: The regression model is trained to minimize the cosine loss between fastText embeddings and image feature vectors.

Additionally, instSift, a cross-platform similarity search library, is used for nearest neighbor (NNS) searches.

Training

Hyperparameter Search and tuning
1. Learning rate: Model training is done with varying it to determine the optimum value.
2. Number of frozen layers: At each stage, 1 is varied and model is fine-tuned.
3. Momentum: 0, 0.85, 0.95. 4. Weight decay 0.01. Optimization: Adam [21] = 0.9, β2 = 0.99

Model Performance
Stage I
Stage II

Observation: Adding a bunch normalization layer in the end helped train the network much faster.

Experiments & Results
Model comparison with baseline

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqueezeNet</td>
<td>88.0%</td>
</tr>
<tr>
<td>InstSift</td>
<td>89.0%</td>
</tr>
</tbody>
</table>

PCA Analysis: PCA analysis after epoch-2 and epoch-8 for a sub-sample, 4 classes.

With more iterations, image vectors begin converging to their fastText equivalents.

Text to Image & Nearest neighbor search in model predictions using fastText embedding for prosoled text

Model to Image & Nearest neighbor search in model predictions using model output for an image [Zero-shot eg on right]

Image to Text & Top k labels for an image based on similarity of its model output to the fastText embeddings for various class labels [example: butterfly, elephant, sheep, bear, hippopotamus, giraffe].

Top-1 Accuracy: In case of class imbalance, average per class top-1 accuracy seems to be a more appropriate choice as described in [3].

Future work involves (1) Improving Zero-shot learning performance of SqueezeNet1.1 model (2) Exploring techniques to deal with class imbalance issue (3) Extending the concept of Semantic Embeddings to Audio datasets.

Conclusion
This project demonstrates that It is feasible to build lightweight Visual-Semantic models for mobile applications while meeting acceptable performance threshold. Applications such as gallery photo search, tag generation, cataloging new products (zero-shot learning) can use make-up of such models.

References
[2] Iandola et al., 2016: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size.