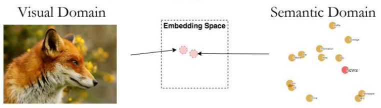
Deep Visual-Semantic Embedding Models for Mobile

Swarna Saraf (email: swsaraf@stanford.edu)

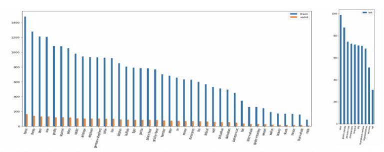
Motivation

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 exploding 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) augment the classification model for many interesting applications as demonstrated in this project.



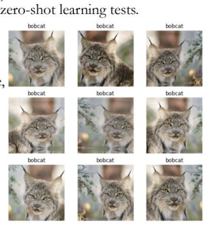
Data



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

Train/valid split: 90:10 of 40 classes (30337 images)
Test set:10 classes (6985 images) are kept aside for zero-shot learning tests.

Data Augmentation techniques help models generalize better!
Pixel and coordinate transforms, such as flip, rotate, warp, zoom, lighting transforms are applied in an optimized way using fastai library [4]

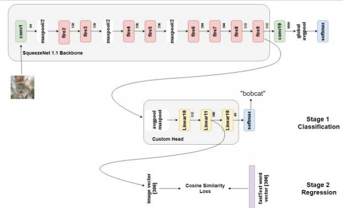


Pre-trained word vectors (300 dimensional) trained using **fastText** are used: 1 million word vectors trained on Wikipedia 2017, UMBC web-base corpus and statmt.org news dataset (16B tokens).

References

[1] Frome et. al, 2013 Devise: A deep visual semantic embedding model
[2] Iandola et. al, 2016 Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size.
[3] Xian et. al, 2017 Zero-shot learning - the good, the bad and the ugly.
[4] Fastai Deep Learning Library. https://github.com/fastai/fastai
[5] Leslie N. Smith, 2018 A Disciplined Approach to Neural Network Hyper-Parameters: Part 1- Learning Rate, Batch Size, Momentum and Weight Decay

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 layers) 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.

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

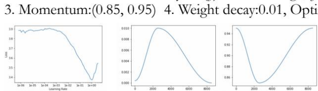
Additionally, **nmslib**, a cross-platform similarity search library, is used for nearest neighbor (kNN) searches.

Training

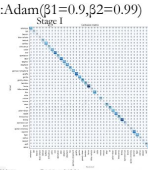
Training Process: **Transfer learning** is used to retrain Squeezenet 1.1 [2] model pre-trained on Imagenet. **1 cycle policy** [5] is used to train network faster.

Hyperparameter Search and tuning

1. Learning rate(α): Mock training is done with varying α to determine the optimum value 2. Number of frozen layers (l_f): At each stage, l_f is varied and model is fine tuned 3. Momentum:(0.85, 0.95) 4. Weight decay:0.01, Optimization:Adam($\beta1$=0.9,$\beta2$=0.99)



1 cycle policy



Model Performance

Stage I

Total time: 14:57

epoch	train_loss	valid_loss	accuracy
1	1.506522	0.700223	0.807297
2	1.201137	0.582587	0.834163
3	0.968867	0.481622	0.863682
4	0.862815	0.421496	0.880265
5	0.746806	0.410386	0.884909

Total time: 09:08

epoch	train_loss	valid_loss	accuracy
1	0.759761	0.405217	0.889221
2	0.816774	0.392443	0.890879
3	0.729095	0.398286	0.890547

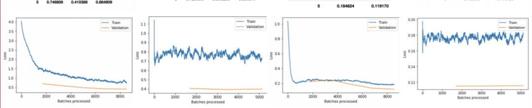
Stage II

Total time: 3:41:33

epoch	train_loss	valid_loss
1	0.246013	0.219739
2	0.234487	0.246009
3	0.230094	0.183701
4	0.194787	0.132013
5	0.184824	0.119170

Total time: 2:15:34

epoch	train_loss	valid_loss
1	0.179190	0.115157
2	0.177406	0.115355
3	0.182653	0.115726



Observation: Adding a batch normalization layer in the end helped train the network much faster!

Experiments & Results

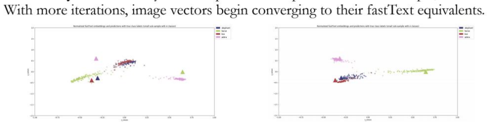
Model comparison with baseline

Stage 1	Total Parameters	Size on Disk(MB)	Epochs	Accuracy
Resnet34	21979164 (100%)	84 (100%)	10	94.66%
Squeezenet 1.1	1416988 (6.45%)	6 (7.14%)	8	89.05%

Stage 2	Total Parameters	Size on Disk(MB)	Epochs	Top1 Accuracy	Top3 Accuracy	Top5 Accuracy	Top10 Accuracy
Resnet34	21967044	84 (100%)	8	74.52%	86.87%	89.82%	93.17%
Squeezenet 1.1	1404868	6 (7.14%)	8	80.60%	89.09%	91.24%	94.30%

Zero-Shot Results (Stage 2)	Top5 Accuracy	Top10 Accuracy
Resnet34	33.56%	53.60%
Squeezenet 1.1	39.13%	53.31%

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: k Nearest neighbor search in model predictions using fastText embedding for provided text

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

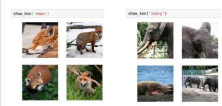
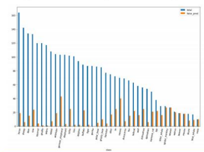


Image to Text: Top k labels for an image based on similarity of its model output to the fastText embeddings for various class labels

```
['gorilla', 'chimpanzee', 'elephant', 'zebra', '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]

$$acc_{\mathcal{Y}} = \frac{1}{\|\mathcal{Y}\|} \sum_{c=1}^{\|\mathcal{Y}\|} \frac{\#\text{ correct predictions in c}}{\#\text{ samples in c}}$$

Top-1 Accuracy result = 67.36%

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 make use of such models,

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.