Sentiment Analysis of images using transfer learning

Presented by: Abhishek Vasudevan

Problem statement

- Social media users often post images with little to no description of the image
- In these cases, text cannot be reliably used to judge sentiment of the user’s posts
- Judge sentiment of post based on the image
- Given an image, determine its sentiment polarity (positive, negative, neutral)

Example images

Positive

Negative

Neutral

Prior work

- Much work has been done for sentiment analysis of text using LSTMs
- Work related to combination of both textual and image data
- Visual sentiment analysis has been explored but these do not employ a large scale dataset

Prior work

- L. Vadicamo *et al*., "Cross-Media Learning for Image Sentiment Analysis in the Wild," *2017 IEEE International Conference on Computer Vision Workshops*

- Extracted images and corresponding text from Twitter and build an image dataset

- Trained a VGG19 on this dataset using transfer learning and use it predict sentiment polarity for images

Project goal

- Can some architecture instead of VGG19 be used to improve performance?
- DenseNet121 is a much deeper architecture compared to VGG19
- Has shown to perform better than VGG
- Hence, this could possibly improve performance for this task
- Goal is to evaluate performance of DenseNet for this task in comparison with VGG19.

Motivation for using DenseNet121

- Alleviate vanishing gradient problem
- Number of parameters less than VGG19
- Improved efficiency in learning: reduced overfitting and hence works on tasks with smaller training set sizes.

Top 1 error from ILSVRC validation dataset

	Top-1	Top-5	10-5	Size	Stem
VGG16	28.732	9.950	8.834	138.4M	14.7M
VGG19	28.744	10.012	8.774	143.7M	20.0M
ResNet50	25.072	7.940	6.828	25.6M	23.6M
ResNet101	23.580	7.214	6.092	44.7M	42.7M
ResNet152	23.396	6.882	5.908	60.4M	58.4M
ResNet50V2	24.040	6.966	5.896	25.6M	23.6M
ResNet101V2	22.766	6.184	5.158	44.7M	42.6M
ResNet152V2	21.968	5.838	4.900	60.4M	58.3M
ResNeXt50	22.260	6.190	5.410	25.1M	23.0M
ResNeXt101	21.270	5.706	4.842	44.3M	42.3M
InceptionV3	22.102	6.280	5.038	23.9M	21.8M
InceptionResNetV2	19.744	4.748	3.962	55.9M	54.3M
Xception	20.994	5.548	4.738	22.9M	20.9M
MobileNet(alpha=0.25)	48.418	24.208	21.196	0.5M	0.2M
MobileNet(alpha=0.50)	35.708	14.376	12.180	1.3M	0.8M
MobileNet(alpha=0.75)	31.588	11.758	9.878	2.6M	1.8M
MobileNet(alpha=1.0)	29.576	10.496	8.774	4.3M	3.2M
MobileNetV2(alpha=0.35)	39.914	17.568	15.422	1.7M	0.4M
MobileNetV2(alpha=0.50)	34.806	13.938	11.976	2.0M	0.7M
MobileNetV2(alpha=0.75)	30.468	10.824	9.188	2.7M	1.4M
MobileNetV2(alpha=1.0)	28.664	9.858	8.322	3.5M	2.3M
MobileNetV2(alpha=1.3)	25.320	7.878	6.728	5.4M	3.8M
MobileNetV2(alpha=1.4)	24.770	7.578	6.518	6.2M	4.4M
DenseNet121	25.028	7.742	6.522	8.1M	7.0M
DenseNet169	23.824	6.824	5.860	14.3M	12.6M

Dataset used for training

- Fortunately, dataset used in base paper available online
- Balanced Twitter for Sentiment Analysis dataset (B-T4SA):
- Equal amount of examples for each sentiment (pos,neu,neg)

Sentiment	**T4SA** *(tweets)*	*(images)*	**T4SA** w/o near-duplicates *(images)*	**B-T4SA** *(images)*
Positive	371,341	501,037	372,904	156,862
Neutral	629,566	757,895	444,287	156,862
Negative	179,050	214,462	156,862	156,862
Sum	904,395	1,473,394	974,053	470,586

B-T4SA dataset

Image	Positive	Neutral	Negative
Image 1	0.90	0.05	0.05
Image 2	0.67	0.23	0.10
Image 3	0.40	0.55	0.5
Image N	…	…	…

Data preprocessing

Source
random 1% of globally produced tweets

Filtering
We keep:
- Original (no retweets)
- English language
- At least 5 words and at least 1 image
- Without other media (no GIFs or vids)

Labelling

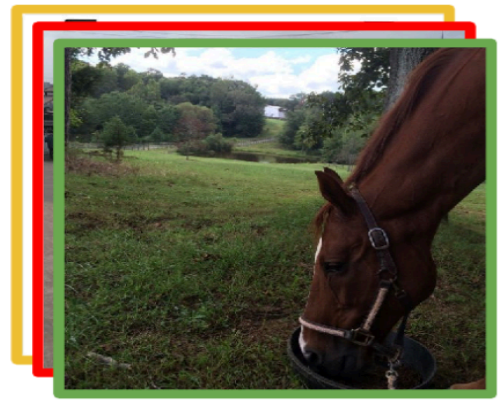
Textual Sentiment Classifier (3-way)

labels and confidences

unlabelled images

Keep tweet if confidence is above threshold

high confidence labels



labeled images

Near-duplicate Detection

T4SA Dataset

Balanced Subset Selection

B-T4SA Dataset

Method used

- Preprocess images from the training dataset
- Feed preprocessed images into DenseNet and train it
- Test and evaluate the model using Twitter Testing data

Preprocessing images

- Resize images into size 224,224,3 which is the input size for the DenseNet
- Convert confidence scores: [0.95, 0.05, 0.05] -> [1, 0, 0]
- Normalize image values to be in range [0,1]
- Form dataset having [<image 1, pos, neu ,neg> , <image 2, pos, neu, neg> … , <img n, pos, neu, neg>]
- Feed this into the DenseNet and train

Problems with data

- Although, the data was well labelled, the problem was with the dataset size:

- Around 63GB: 470,586 images

- Used portion of data for training my model:
  - Used tensorflow’s tf.data library to extract portion of data
  - Used this small portion to train the model

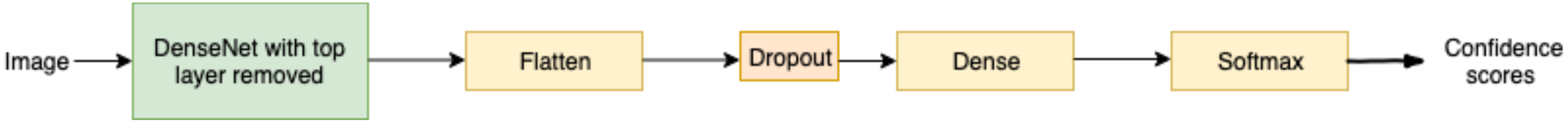
Training data used in experiments

- 50k images were extracted from the original 470,586 images
- Balanced number of examples for each class was used:
  - Positive: 15,441
  - Negative: 14,887
  - Neutral: 15,287
- Validation set of 5,000 images were chosen at random

Training DenseNet

- Tensorflow makes it easy to import pretrained models and retrain it for our own purposes.
- Loaded DenseNet with weights trained from imagenet
- Freeze all these pretrained layers
- Output layer of DenseNet is removed
- Following layers are added:
  - Flatten
  - Dropout (0.3)
  - Dense layer
  - Softmax for 3 output classes

Flow diagram



Training details

- 20 epochs used for training DenseNet121 with step size of 500 for each epoch
- Adam optimizer
- Learning rate of 10^{-5} was used
- L2 regularization to prevent overfitting
- Categorical cross entropy

Twitter Testing dataset

- Twitter testing dataset
- 1269 images in total
- Labelled as positive or negative by AMT workers

Sentiment	Twitter Testing Dataset		
	5 agree	$\geq$ 4 agree	$\geq$ 3 agree
Positive	581	689	769
Negative	301	427	500
Sum	882	1,116	1,269

Evaluation

- Model was trained to predict for three classes (positive, neutral, negative), but test dataset has only two classes (positive, negative)
- Choose output as either positive or negative based on softmax output
- Prediction accuracy is the evaluation metric

Evaluation

- Would be unfair to compare the trained DenseNet with the VGG given in base paper:

- VGG in base paper was trained on the whole dataset, while DenseNet was trained on only around 10% of this data

- So, I trained my own VGG model on this 10% data to make a fair comparison

Results

Model used	**5 agree**	**>= 4 agree**	**>= 3 agree**
VGG19 on 10% data	61.678	59.946	58.392
DenseNet121 on 10% data	64.62	62.00	60.75
VGG19 on whole dataset – fine tuned (base paper)	78.5	75.5	72.5

Discussion

- Only 10% of the original training dataset was used
- In such a small data sample, DenseNet achieved 2-3% better accuracy than VGG
- Fine tuning will possibly improve accuracy

Problems faced

- Initially, planned to implement ResNet50 instead of DenseNet121

- Was not able to train Resnet even after tweaking various hyperparameters
- Turns out several people have had this issue in the past: https://github.com/keras-team/keras/issues/9214

- Caused because of the way Batch Normalization was implemented in Keras

- Solution was to unfreeze certain layers and train

Example predictions

Ground truth: Negative
Prediction: Negative

Ground truth: Positive
Prediction: Positive

Incorrect predictions

Ground truth: Negative
Prediction: Positive

Ground truth: Positive
Prediction: Negative

Future work

- Using the entire dataset of 470,586 images for training the DenseNet model
- Fine-tuning the pre-trained layers by unfreezing them
- Evaluate on images obtained from Instagram

Conclusion

- Built a DenseNet model for visual sentiment analysis
- Evaluated performance for DenseNet and VGG
- DenseNet proved to outperform VGG on the small dataset used
- Suggested improvements which could be made

Thank you!