Team 55

Predict attribute labels for restaurants to write fake reviews

Arda Özdere Haydar Şahin Sebastian Muhle

Introduction

We want to compare the performance of the Inception v3 architecture to the performance of the Xception architecture. As Francois Chollet showed in his Xception paper, while only marginally better on ImageNet, Xception was 4.3% better on Google's internal dataset JFT. He suggests that "This may be due to the fact that Inception V3 was developed with a focus on ImageNet and may thus be by design over-fit to this specific task." In our project, we want to use the Kaggle Yelp Restaurant Photo Classification to compare both architectures and see how big the performance gap is on this task. Additionally, we want to use the generated multiple labels to write fake restaurant reviews using an RNN trained on a dataset of Yelp reviews.

Related Work

The Kaggle Yelp Restaurant Photo Classification was a Kaggle Competition so there already exist benchmarks. Since the challenge was finished in April 2016, the participants only used ResNet and Inception v3 architectures. We didn't find any later examples on the internet of people using the Xception architecture on this task.

Researchers from the University of Chicago have already written a paper on how to write very convincing fake reviews for restaurants using RNNs. In their method, they used specific metadata about the restaurant like the name of their dishes to produce the fake reviews. With our approach, we want to use the generated multiple labels from our CNN to write these reviews. In this way, we hope we can use image data about the restaurant to make the fake reviews even more convincing.

Datasets

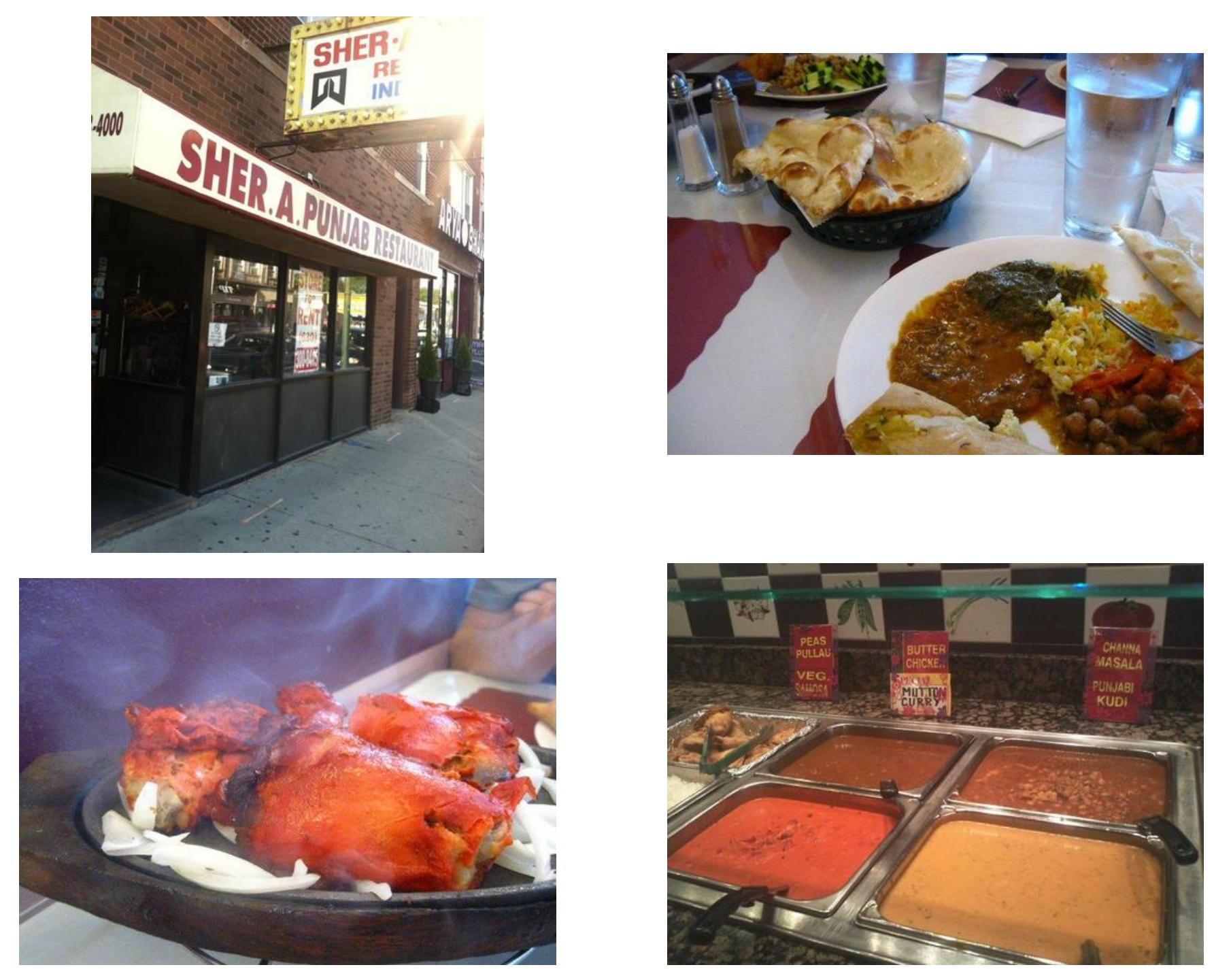
Yelp Restaurant Photos:
- **~235k images** of 2000 different restaurants
- Average pics per restaurant ~ 117,5; Range from 1 to 300
- **Multi-label classification** problem with **9 different labels**
- 0: good for lunch; 1: good for dinner; 2: takes reservations; 3: outdoor seating; 4: is expensive; 5: has alcohol; 6: has table service; 7: ambience is classy; 8: good for kids

Yelp Reviews:
- **2.225k reviews** for 77.5k businesses written by 500k different users
- Several different languages

"Awesome meal. Very willing to make adjustments and recommendations. Would come back here in a heartbeat."

"Love the veggie patties, when they are available...haven't really eaten anything else there."

"By far the best burrito and salsa I have ever had. This place is way across town from where I live but i go there just to get my burritos, yes if I'm gonna go across town I better get two burritos to go."



Business ID: 1001 Labels: 0 1 6 8

Business ID: 1000 Labels: 1 2 3 4 5 6 7

Methodology

CNN - Photo Classification

Preprocessing:
- We mapped the restaurants and their attributes to the induvial image to label them.
- For hyperparameter research, we used a **subset** of **60k training images, 10k validation images** -> saved in an HDF5 file
- **Used 70k learning images & 235k test images**
- We relied on **Image Augmentation** (rotating, zooming, horizontal flipping) and **applied the mean** to the data.
- We wrote a **multi-label generator** to use the data in Keras.

Architecture:
- To **test our code** we used **transfer learning** on a **VGG16** architecture with **pretrained image net weights**.
- For the **Inception v3** and the **Xception** architecture, we also used **pretrained** model from Keras with **image net weights**.
- For all models, we **replaced the top** with a global average pooling layer, followed by a fully connected layer (size of 1024 and a relu activation function) and a prediction layer (size of 9 and a **sigmoid activation function for multi-label classification**).
- To classify businesses, we have averaged predictions of the images of business

Training:
- To test our preprocessing we first trained with a very small subset on VGG16 and later with a bigger one until we achieved promising results.
- We used **Adam** as our optimizer.
- For **multi-label classification**, we used a **binary-cross entropy loss** and an **F1-score** as our metric.
- For hyperparameter search, we used a **grid search** and tried different learning rates and numbers of freezed layers.
- We used checkpoints during training.

$$F1 = 2\frac{p\times r}{p+r} \text{ where p} = \frac{tp}{tp+fp}, r = \frac{tp}{tp+fn}$$

F1-score

RNN – Fake Review Generation

Preprocessing:
- After importing the dataset, we created a set of all of the distinct characters in the text and then create a **map of each character to a unique integer**. We did this to train the model on integers.
- To define the training data, we split the text into **subsequences of 100 characters**.
- We created these sequences by sliding along the whole text one character at a time, allowing each character a chance to be learned from the 100 characters that preceded it (except the first 100 characters). While doing this, we convert the characters to integers using our lookup table we prepared earlier.
- We transformed the input sequences into the form of [samples, time steps, features] so that they can fit into an LSTM.
- We **normalized** the input.
- We converted the **output** patterns into a **one hot encoded vector**. That is, that the network can predict the probability of the different character in our lookup table.

Architecture:
- For our RNN we used **LSTMs layers** with 256 memory units.
- Every LSTM layer is followed by a **dropout layer**.
- The **prediction** layer uses a **softmax** activation function.

Training:
- For hyperparameter search, we used a **grid search**. We tried various numbers of LSTM and dropout layer pairs from 2 - 6, different dropout rates and learning rates.
- We used **Adam** as our optimizer.
- We used a **categorical cross entropy loss**.
- We used checkpoints during training.
- We used only **xx% of the training set**.

Text generation:
- To generate a fake review we first check which predicted labels from the CNN are above a certain threshold.
- **For every label** above it, we feed a **prepared seed sequence** into the model as an input.
- The model then **updates the seed sequence** to add the generated character on the end and trim off the first character.
- In the end, we **concatenate the different generated sequences** to one restaurant review.
- Notice that you have to convert the seed sequence into integers and the output into characters.

Outcome

CNN - Photo Classification:
- **Our** best F1-score: **0.69**
- **Winner** F1-score: **0.83** , we are place 155 of 355
- **Inception v3 vs Xception**: In our case, **both performed quite similar**. Further research is necessary to find out if one of the architectures plateaus as we are getting closer to a 0.83 score

RNN – Fake Review Generation:
- As a metric, some of our friends classified 20 reviews. 10 were real 10 were fake.
- Xx of our fake reviews were considered to be real and xx of the real reviews were considered to be real.

Picture 1 results

Picture 2 results

"The service was a botp of the seally aod the service was a bat of the seally aod the service was a little of the seally aod the seally aod the seally aod the service was a little oi the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the seally aod the seally aod the service was a little of the"

RNN result

Learnings

- **VGG16** is indeed **great for prototyping** and it was easier to achieve good results with it than with the Inception v3 architecture or the Xception architecture.
- Using **bottlenecks** can really speed up training. However, memory issues can occur and you have to fix them.
- **Batch size** is important for training speed. We found out that a batch size of **16** works best for us when training an Inception v3 model on a K80.

Ideas for improvement:
- Using **class weights** to improve the balance of the photo dataset
- Use the **full datasets** for the CNNs and the RNN.
- Filter out the few reviews which were written in other languages to improve the RNN results.
- Sample the characters in the text generation instead of using the one with the max. probability to get more diversity.
- Use **better seed sequences** for the RNN to write more convincing fake reviews. One idea would be to have not one but a few different seed sequences for every predicted label and then randomly choose one. Also, you could shuffle the order of the predictions for the labels, so that label 0 isn't always the first sequence in a review.
- Use a **GAN to conditional generate text** based on the predicated labels to write more natural sounding fake reviews.
- Use a more advanced approach for the multi-instance approach of this problem

Our GitHub Repository