

Automatic Understanding and Classification of Image Advertisements

Computer Vision,
Northeastern University
Alesya and Divya

Outline

- Advertisements and their applications
- Recent work on understanding advertisements
 - Problem
 - Dataset
 - Experiments
- Our project goals

Understanding Advertisements

- Convey message to viewer
- Ads may include images and/or text
- Ads may use visual rhetoric:
 - Symbolism
 - Inference
 - Non-photorealistic objects



Applications

- Predict how effective a certain ad will be
- Better ad targeting according to user interests
- Source of revenue for companies
- Motivate the user to buy/try the product

Recent work

A recent CVPR conference paper:

"Automatic Understanding of Image and Video Advertisements" (CVPR 2017), Hussain et al.

- Link to the paper-
http://people.cs.pitt.edu/~kovashka/hussain_zhang_kovashka_ads_cvpr2017.pdf
- Proposes a novel problem of understanding advertisements
- Introduces a newly created dataset to tackle this problem
- Include baseline classification results from initial experiments

Why ads?

- Why is this an important problem?
 - Part of our culture
 - Ads generate revenue
- Why is it interesting?
 - Ads convey a persuasive message
 - Many ads require decoding of their meaning
 - What, how and why



Why ads?

- Why is this an important problem?
 - Integral part of our culture
 - Ads generate revenue
- Why is it interesting?
 - Ads convey a persuasive message
 - Many ads require decoding of their meaning
 - What, why, and how



Why ads?

- Why is this an important problem?
 - Integral part of our culture
 - Ads generate revenue
- Why is it interesting?
 - Ads convey a persuasive message
 - Many ads require decoding of their meaning
 - What, why, and how

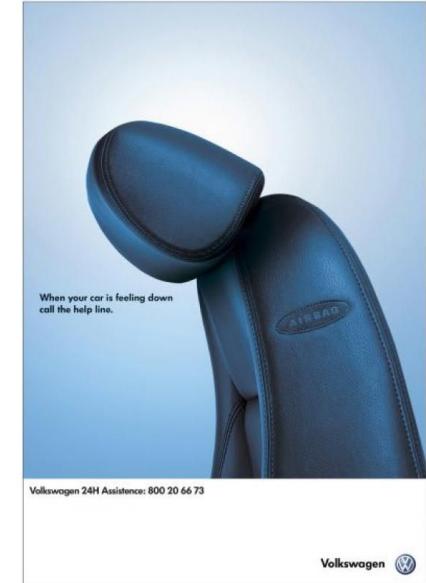


Dataset

- 64,832 image ads
- Product ads and public service announcements (PSA)
- Annotations:
 - Product or subject of the ad
 - Sentiment attempts to inspire
 - Strategy used to convey message
 - Symbolism used
 - Answers to questions:
 - “What should the viewer do according to this ad?”
 - “Why should he/she do it?”

Initial Experiments

- Topic classification
- Sentiment classification
- Symbolism prediction model
- Question-answering

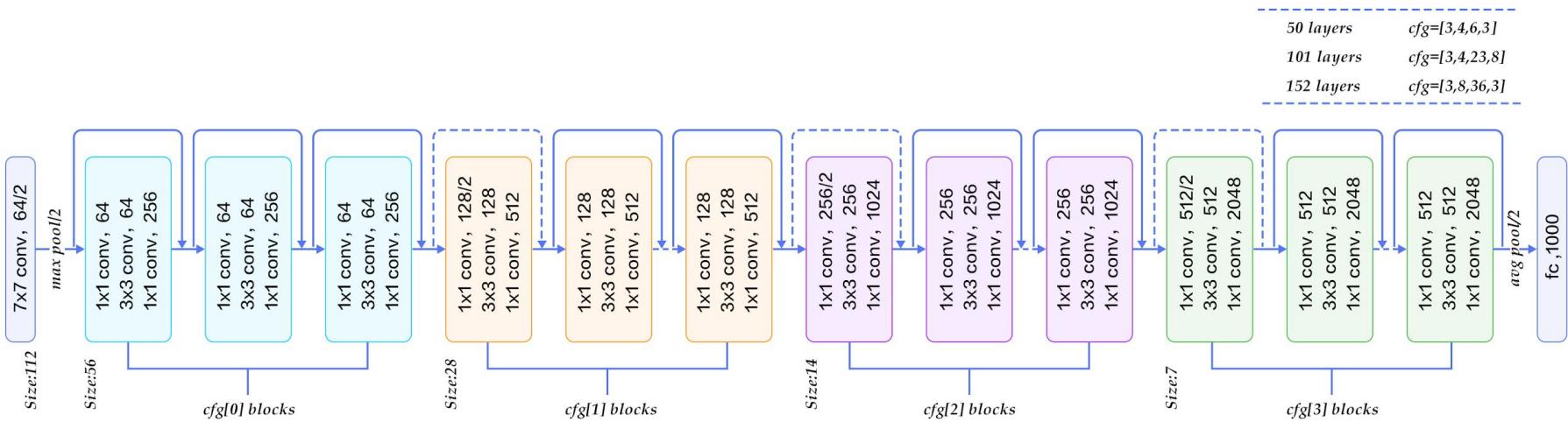


Topic and Sentiment: Dataset Annotations

Topic	Sentiment
Restaurants, cafe, fast food	Active (energetic, etc.)
Coffee, tea	Alarmed (concerned, etc.)
Sports equipment, activities	Amazed (excited, etc.)
Phone, TV and web providers	Angry (annoyed, irritated)
Education	Cheerful (delighted, etc.)
Beauty products	Disturbed (disgusted, shocked)
Cars, automobiles	Educated (enlightened, etc.)
Political candidates	Feminine (womanly, girlish)
Animal rights, animal abuse	Persuaded (impressed, etc.)
Smoking, alcohol abuse	Sad (depressed, etc.)

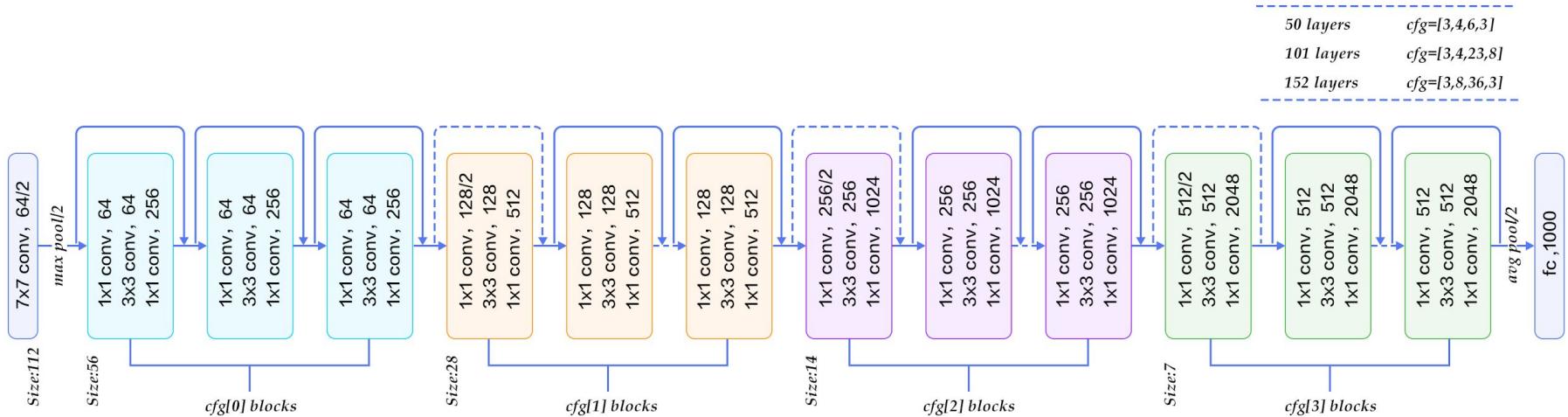
Table 2: A sample from our list of topics and sentiments.
See supp for the full list of 38 topics and 30 sentiments.

Topic and sentiment: Experiment



ResNet - 152 layers

Topic and sentiment: Experimental Results



38 topics and 30 sentiments
Topics - 60.34% accuracy
Sentiment - 27.92% accuracy

Proposed Goals

- MUST goal: Use transfer learning on a new architecture (VGG), and compare performance to ResNet model results in the paper
- Bonus goal : Compare how the model works using Random Decision Forests.

Ads Dataset

Steps

1. Load the dataset onto Maria's Machine
2. Preprocess the Image data
3. Preprocess the labels
4. Train on topic classification

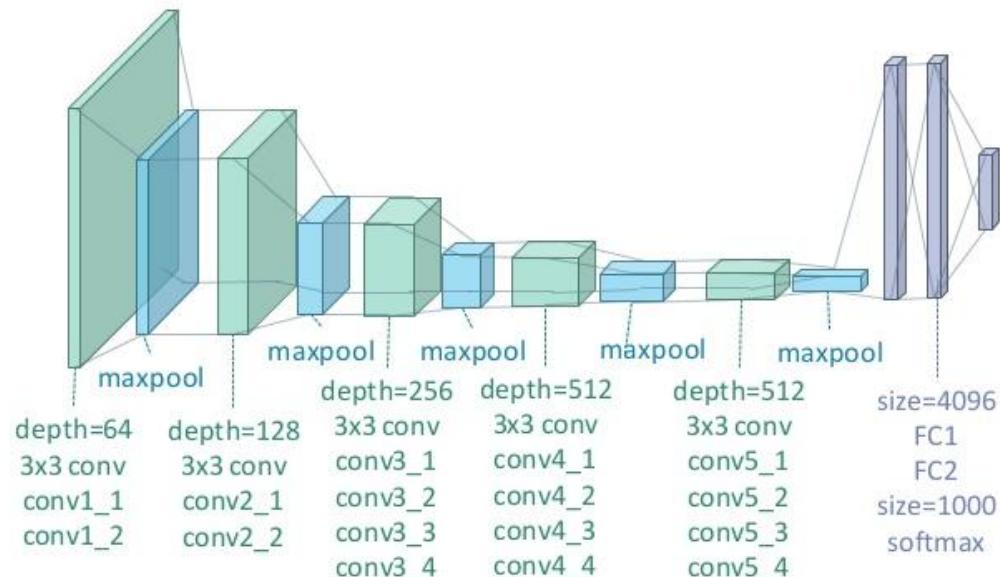
Data Preprocessing

Steps

1. Preprocess the Image data
 - a. VGG preprocessing with TensorFlow Slim library
 - b. Resize 224 x 224
 - c. Normalize
2. Preprocess the labels
 - a. Select most common label (out of 3 annotations)
 - b. Select one of the annotations at random

Architecture - VGG

VGG 19



Replace output layer
with ad classes (39 total)
and finetune network

RMSProp Optimizer with Learning Rate = 0.0001

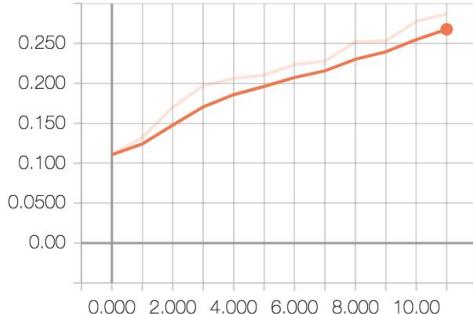
```
Start Time is : 2017-12-12 22:35:06.799926
Train on 3766 samples, validate on 772 samples
Epoch 1/12
3766/3766 [=====] - 105s 28ms/step - loss: 3.3279 - acc: 0.1179 - val_loss: 3.2206 - val_acc: 0.0972
Epoch 2/12
3766/3766 [=====] - 103s 27ms/step - loss: 3.0975 - acc: 0.1277 - val_loss: 2.9937 - val_acc: 0.1619
Epoch 3/12
3766/3766 [=====] - 103s 27ms/step - loss: 3.0156 - acc: 0.1726 - val_loss: 3.0480 - val_acc: 0.1464
Epoch 4/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.9305 - acc: 0.1997 - val_loss: 2.9531 - val_acc: 0.1865
Epoch 5/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.8858 - acc: 0.2116 - val_loss: 2.8148 - val_acc: 0.2435
Epoch 6/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.8313 - acc: 0.2265 - val_loss: 2.8418 - val_acc: 0.2409
Epoch 7/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.8195 - acc: 0.2326 - val_loss: 2.8944 - val_acc: 0.2474
Epoch 8/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.7857 - acc: 0.2422 - val_loss: 2.8304 - val_acc: 0.2396
Epoch 9/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.7700 - acc: 0.2377 - val_loss: 2.9050 - val_acc: 0.2124
Epoch 10/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.7270 - acc: 0.2472 - val_loss: 2.8254 - val_acc: 0.2565
Epoch 11/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.7264 - acc: 0.2507 - val_loss: 2.9046 - val_acc: 0.2383
Epoch 12/12
3766/3766 [=====] - 103s 27ms/step - loss: 2.7056 - acc: 0.2618 - val_loss: 3.0451 - val_acc: 0.1930
('End Time is : ', datetime.datetime(2017, 12, 12, 22, 55, 53, 36343))
('Total time is :', datetime.timedelta(0, 1246, 236417))
1135/1135 [=====] - 9s 8ms/step
('Test loss:', 3.0687704319470779)
('Test accuracy:', 0.1991189429675955)
```

Adam Optimizer with Learning Rate = 0.0001

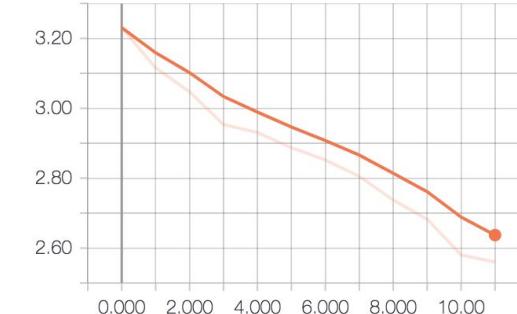
```
Epoch 2/15
3766/3766 [=====] - 108s 29ms/step - loss: 3.0815 - acc: 0.1585 - val_loss: 3.0052 - val_acc: 0.1645
Epoch 3/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.9820 - acc: 0.1986 - val_loss: 2.9805 - val_acc: 0.1749
Epoch 4/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.9093 - acc: 0.2135 - val_loss: 2.8907 - val_acc: 0.2085
Epoch 5/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.8564 - acc: 0.2143 - val_loss: 2.8937 - val_acc: 0.1982
Epoch 6/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.7947 - acc: 0.2334 - val_loss: 2.9141 - val_acc: 0.1917
Epoch 7/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.7448 - acc: 0.2435 - val_loss: 2.8510 - val_acc: 0.2124
Epoch 8/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.6910 - acc: 0.2520 - val_loss: 3.0387 - val_acc: 0.1632
Epoch 9/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.6560 - acc: 0.2637 - val_loss: 2.8100 - val_acc: 0.2370
Epoch 10/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.5787 - acc: 0.2831 - val_loss: 2.8412 - val_acc: 0.2306
Epoch 11/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.5077 - acc: 0.3056 - val_loss: 2.8120 - val_acc: 0.2539
Epoch 12/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.4449 - acc: 0.3197 - val_loss: 2.7898 - val_acc: 0.2604
Epoch 13/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.3418 - acc: 0.3542 - val_loss: 2.9007 - val_acc: 0.2241
Epoch 14/15
3766/3766 [=====] - 108s 29ms/step - loss: 2.1961 - acc: 0.3720 - val_loss: 2.9365 - val_acc: 0.2526
Epoch 15/15
3766/3766 [=====] - 108s 29ms/step - loss: 1.9915 - acc: 0.4376 - val_loss: 3.0221 - val_acc: 0.2539
('End Time is : ', datetime.datetime(2017, 12, 12, 19, 2, 25, 238472))
('Total time is :', timedelta(0, 1630, 607568))
Saved trained model at /home/micha/project/model4/saved_models/model4_trained.h5
2017-12-12 19:02:26.703793: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
3.45GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
1135/1135 [=====] - 9s 8ms/step
('Test loss:', 3.0253931039230415)
('Test accuracy:', 0.2528634363596602)
```

Adam Optimizer with Learning Rate = 0.0001

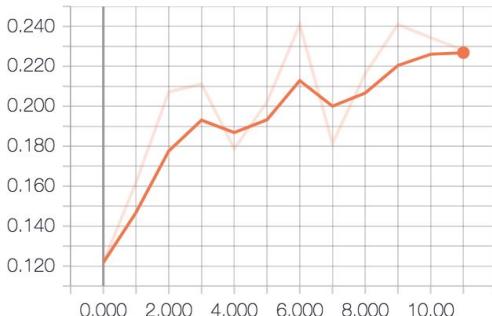
acc



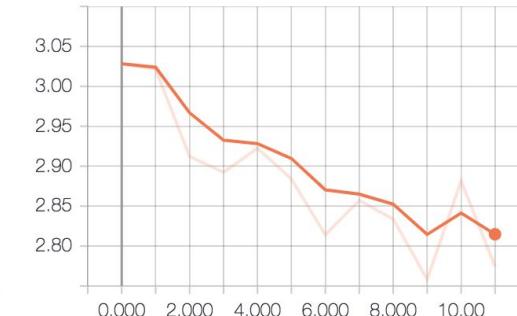
loss



val_acc



val_loss

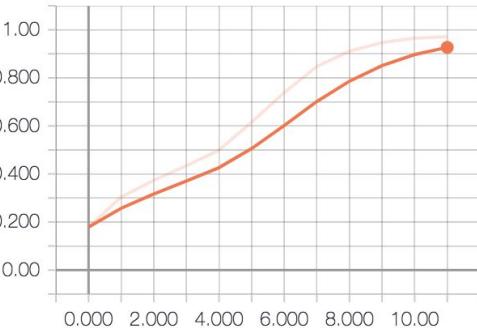


Adam Optimizer with Learning Rate = 0.00001

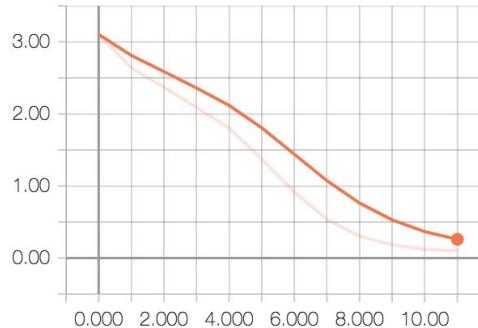
```
Epoch 1/12
2017-12-12 21:19:23.330449: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
2.20GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
2017-12-12 21:19:23.552824: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
2.20GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
2017-12-12 21:19:23.658456: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
2.20GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
2017-12-12 21:19:23.701682: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
2.15GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
3766/3766 [=====] - 111s 30ms/step - loss: 3.1024 - acc: 0.1792 - val_loss: 2.7488 - val_acc: 0.2927
Epoch 2/12
3766/3766 [=====] - 109s 29ms/step - loss: 2.6389 - acc: 0.3040 - val_loss: 2.5415 - val_acc: 0.3446
Epoch 3/12
3766/3766 [=====] - 109s 29ms/step - loss: 2.3705 - acc: 0.3736 - val_loss: 2.5275 - val_acc: 0.3549
Epoch 4/12
3766/3766 [=====] - 109s 29ms/step - loss: 2.0919 - acc: 0.4341 - val_loss: 2.4528 - val_acc: 0.3705
Epoch 5/12
3766/3766 [=====] - 109s 29ms/step - loss: 1.8028 - acc: 0.4992 - val_loss: 2.4536 - val_acc: 0.3795
Epoch 6/12
3766/3766 [=====] - 109s 29ms/step - loss: 1.3769 - acc: 0.6152 - val_loss: 2.6018 - val_acc: 0.3653
Epoch 7/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.9188 - acc: 0.7384 - val_loss: 3.0846 - val_acc: 0.3562
Epoch 8/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.5369 - acc: 0.8471 - val_loss: 3.2896 - val_acc: 0.3640
Epoch 9/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.3051 - acc: 0.9108 - val_loss: 3.5646 - val_acc: 0.3303
Epoch 10/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.1839 - acc: 0.9472 - val_loss: 3.9702 - val_acc: 0.3225
Epoch 11/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.1238 - acc: 0.9655 - val_loss: 4.5174 - val_acc: 0.3446
Epoch 12/12
3766/3766 [=====] - 109s 29ms/step - loss: 0.0989 - acc: 0.9713 - val_loss: 4.2416 - val_acc: 0.3290
('End Time is : ', datetime.datetime(2017, 12, 12, 21, 41, 14, 922509))
('Total time is :', datetime.timedelta(0, 1317, 157020))
2017-12-12 21:41:15.056716: W tensorflow/core/common_runtime/bfc_allocator.cc:217] Allocator (GPU_0_bfc) ran out of memory trying to allocate
3.45GiB. The caller indicates that this is not a failure, but may mean that there could be performance gains if more memory is available.
1135/1135 [=====] - 9s 8ms/step
('Test loss:', 3.936884893089641)
('Test accuracy:', 0.35330396523034518)
```

Adam Optimizer with Learning Rate = 0.00001

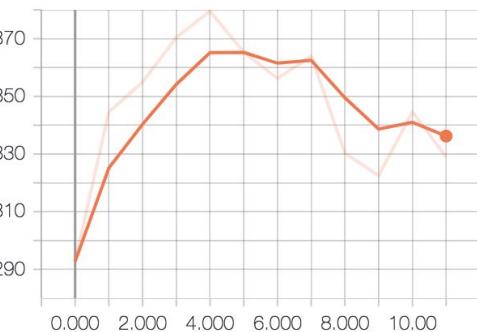
acc



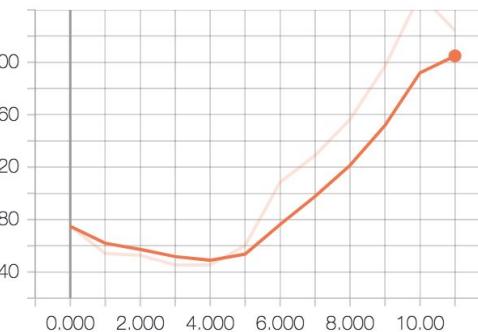
loss



val_acc



val_loss



Stochastic Gradient Descent Optimizer with Learning Rate= 0.00001

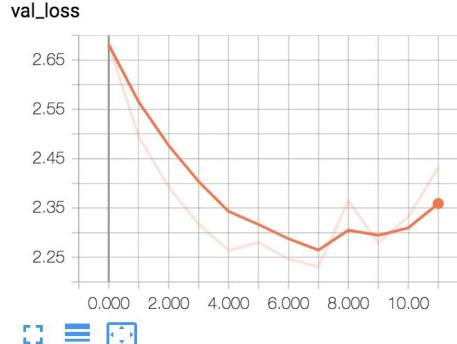
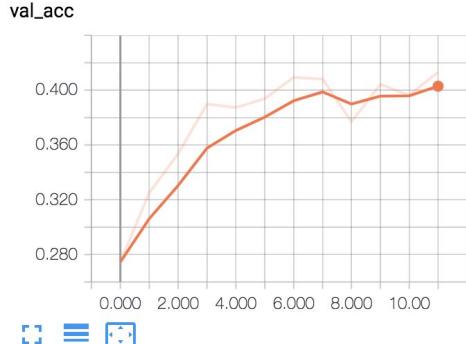
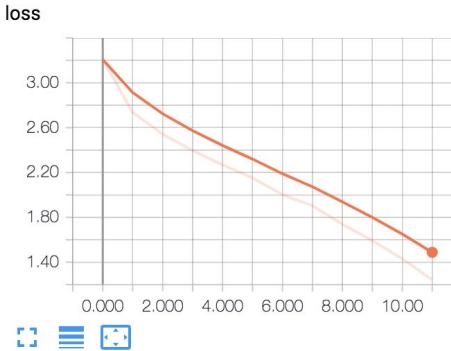
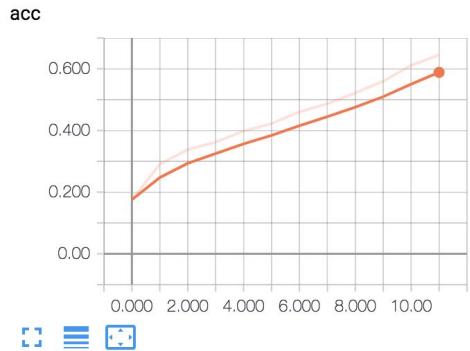
```
Train on 3766 samples, validate on 772 samples
Epoch 1/12
3766/3766 [=====] - 101s 27ms/step - loss: 3.6982 - acc: 0.0929 - val_loss: 2.9960 - val_acc: 0.1891
Epoch 2/12
3766/3766 [=====] - 98s 26ms/step - loss: 3.3032 - acc: 0.1267 - val_loss: 2.9586 - val_acc: 0.2228
Epoch 3/12
3766/3766 [=====] - 99s 26ms/step - loss: 3.2025 - acc: 0.1426 - val_loss: 2.9051 - val_acc: 0.2034
Epoch 4/12
3766/3766 [=====] - 99s 26ms/step - loss: 3.1278 - acc: 0.1569 - val_loss: 2.8598 - val_acc: 0.2396
Epoch 5/12
3766/3766 [=====] - 99s 26ms/step - loss: 3.0382 - acc: 0.1880 - val_loss: 2.8004 - val_acc: 0.2578
Epoch 6/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.9609 - acc: 0.2130 - val_loss: 2.7421 - val_acc: 0.2772
Epoch 7/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.9267 - acc: 0.2302 - val_loss: 2.6947 - val_acc: 0.2824
Epoch 8/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.8517 - acc: 0.2454 - val_loss: 2.6448 - val_acc: 0.3070
Epoch 9/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.7993 - acc: 0.2589 - val_loss: 2.6187 - val_acc: 0.3135
Epoch 10/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.7676 - acc: 0.2716 - val_loss: 2.5723 - val_acc: 0.3238
Epoch 11/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.7014 - acc: 0.2804 - val_loss: 2.5512 - val_acc: 0.3290
Epoch 12/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.6746 - acc: 0.2942 - val_loss: 2.5327 - val_acc: 0.3238
('End Time is : ', datetime.datetime(2017, 12, 13, 1, 22, 32, 796516))
('Total time is :', datetime.timedelta(0, 1189, 996458))
1135/1135 [=====] - 9s 8ms/step
('Test loss:', 2.6412126566344947)
('Test accuracy:', 0.30484581521429155)
```

Stochastic Gradient Descent Optimizer with Learning Rate= 0.0001

```
device: 0, name: GeForce GTX 1080, pci bus id: 0000:01:00.0)
Start Time is : 2017-12-12 23:04:37.244193
Train on 3766 samples, validate on 772 samples
Epoch 1/12
3766/3766 [=====] - 100s 27ms/step - loss: 3.2076 - acc: 0.1753 - val_loss: 2.6811 - val_acc: 0.2746
Epoch 2/12
3766/3766 [=====] - 98s 26ms/step - loss: 2.7367 - acc: 0.2902 - val_loss: 2.4951 - val_acc: 0.3251
Epoch 3/12
3766/3766 [=====] - 98s 26ms/step - loss: 2.5425 - acc: 0.3378 - val_loss: 2.3923 - val_acc: 0.3536
Epoch 4/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.3965 - acc: 0.3622 - val_loss: 2.3179 - val_acc: 0.3899
Epoch 5/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.2682 - acc: 0.3980 - val_loss: 2.2637 - val_acc: 0.3873
Epoch 6/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.1517 - acc: 0.4222 - val_loss: 2.2801 - val_acc: 0.3938
Epoch 7/12
3766/3766 [=====] - 99s 26ms/step - loss: 2.0026 - acc: 0.4602 - val_loss: 2.2465 - val_acc: 0.4093
Epoch 8/12
3766/3766 [=====] - 99s 26ms/step - loss: 1.9034 - acc: 0.4867 - val_loss: 2.2310 - val_acc: 0.4080
Epoch 9/12
3766/3766 [=====] - 99s 26ms/step - loss: 1.7396 - acc: 0.5218 - val_loss: 2.3650 - val_acc: 0.3769
Epoch 10/12
3766/3766 [=====] - 99s 26ms/step - loss: 1.5930 - acc: 0.5597 - val_loss: 2.2790 - val_acc: 0.4041
Epoch 11/12
3766/3766 [=====] - 99s 26ms/step - loss: 1.4305 - acc: 0.6110 - val_loss: 2.3323 - val_acc: 0.3964
Epoch 12/12
3766/3766 [=====] - 99s 26ms/step - loss: 1.2432 - acc: 0.6452 - val_loss: 2.4322 - val_acc: 0.4132
('End Time is : ', datetime.datetime(2017, 12, 12, 23, 24, 26, 957728))
('Total time is : ', datetime.timedelta(0, 1189, 713535))
1135/1135 [=====] - 9s 8ms/step
('Test loss:', 2.5353376791865814)
('Test accuracy:', 0.40176211472124779)
```

Best Model

SGD with Learning rate of 0.0001



Best Model

SGD Optimizer on VGG19 with Learning Rate = 0.001

```
('End time is . . , datetime.datetime(2017, 12, 12, 23, 24, 20, 957720))  
('Total time is :', datetime.timedelta(0, 1189, 713535))  
1135/1135 [=====] - 9s 8ms/step  
('Test loss:', 2.5353376791865814)  
('Test accuracy:', 0.40176211472124779)
```

Training summary

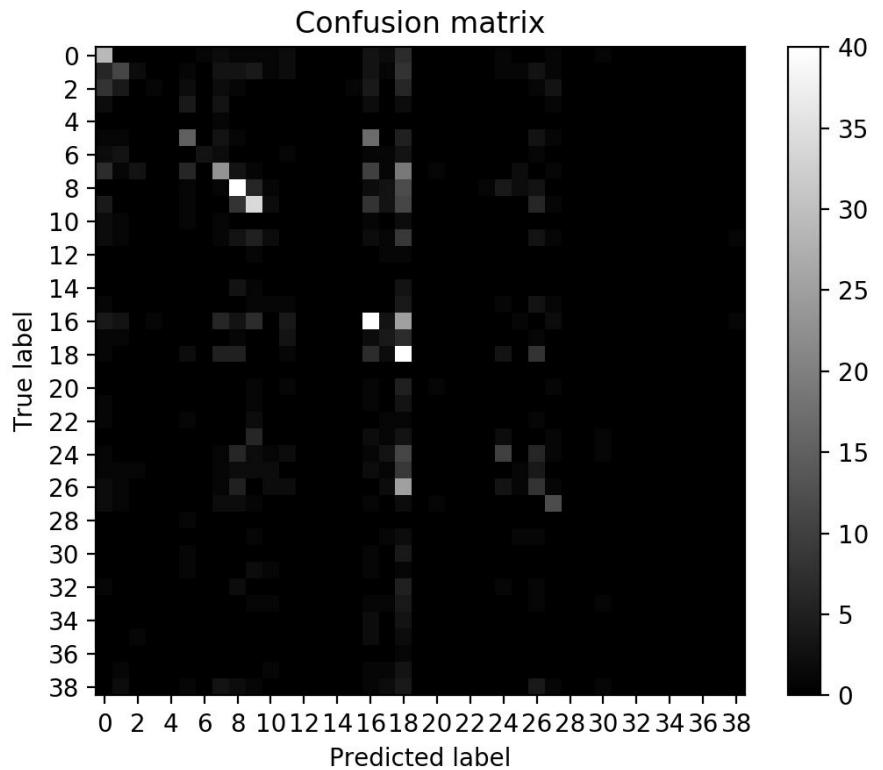
Optimizer	Learning rate	Validation accuracy
RMSProp	.0001	.1930
Adam	.0001	.2539
Adam	.00001	.3290
SGD	.00001	.3238
SGD	.0001	.4132

Results

Table of Comparison

<u>Our Model</u>		<u>Paper Model</u>	
Training Accuracy	64.52	Training Accuracy	N/A
Test Accuracy	40.173%	Test Accuracy	60.34%
Model Used	VGG19	Model Used	ResNet 152-layer
Dataset size	5673	Dataset size	64832

Results: Confusion Matrix



Classes

- 0 "Restaurants, cafe, fast food" (ABBREVIATION: "restaurant")
- 1 "Chocolate, cookies, candy, ice cream" (ABBREVIATION: "chocolate")
- 2 "Chips, snacks, nuts, fruit, gum, cereal, yogurt, soups" (ABBREVIATION: "chips")
- 3 "Seasoning, condiments, ketchup" (ABBREVIATION: "seasoning")
- 4 "Pet food" (ABBREVIATION: "petfood")
- 5 "Alcohol" (ABBREVIATION: "alcohol")
- 6 "Coffee, tea" (ABBREVIATION: "coffee")
- 7 "Soda, juice, milk, energy drinks, water" (ABBREVIATION: "soda")
- 8 "Cars, automobiles (car sales, auto parts, car insurance, car repair, gas, motor oil, etc.)" (ABBREVIATION: "cars")
- 9 "Electronics (computers, laptops, tablets, cellphones, TVs, etc.)" (ABBREVIATION: "electronics")
- 10 "Phone, TV and internet service providers" (ABBREVIATION: "phone_tv_internet_providers")
- 11 "Financial services (banks, credit cards, investment firms, etc.)" (ABBREVIATION: "financial")
- 12 "Education (universities, colleges, kindergarten, online degrees, etc.)" (ABBREVIATION: "education")
- 13 "Security and safety services (anti-theft, safety courses, etc.)" (ABBREVIATION: "security")
- 14 "Software (internet radio, streaming, job search website, grammar correction, travel planning, etc.)" (ABBREVIATION: "software")
- 15 "Other services (dating, tax, legal, loan, religious, printing, catering, etc.)" (ABBREVIATION: "other_service")
- 16 "Beauty products and cosmetics (deodorants, toothpaste, makeup, hair products, laser hair removal, etc.)" (ABBREVIATION: "beauty")
- 17 "Healthcare and medications (hospitals, health insurance, allergy, cold remedy, home tests, vitamins)" (ABBREVIATION: "healthcare")
- 18 "Clothing and accessories (jeans, shoes, eye glasses, handbags, watches, jewelry)" (ABBREVIATION: "clothing")
- 19 "Baby products (baby food, sippy cups, diapers, etc.)" (ABBREVIATION: "baby")
- 20 "Games and toys (including video and mobile games)" (ABBREVIATION: "game")
- 21 "Cleaning products (detergents, fabric softeners, soap, tissues, paper towels, etc.)" (ABBREVIATION: "cleaning")
- 22 "Home improvements and repairs (furniture, decoration, lawn care, plumbing, etc.)" (ABBREVIATION: "home_improvement")
- 23 "Home appliances (coffee makers, dishwashers, cookware, vacuum cleaners, heaters, music players, etc.)" (ABBREVIATION: "home_appliance")
- 24 "Vacation and travel (airlines, cruises, theme parks, hotels, travel agents, etc.)" (ABBREVIATION: "travel")
- 25 "Media and arts (TV shows, movies, musicals, books, audio books, etc.)" (ABBREVIATION: "media")
- 26 "Sports equipment and activities" (ABBREVIATION: "sports")
- 27 "Shopping (department stores, drug stores, groceries, etc.)" (ABBREVIATION: "shopping")
- 28 "Gambling (lotteries, casinos, etc.)" (ABBREVIATION: "gambling")
- 29 "Environment, nature, pollution, wildlife" (ABBREVIATION: "environment")
- 30 "Animal rights, animal abuse" (ABBREVIATION: "animal_right")
- 31 "Human rights" (ABBREVIATION: "human_right")
- 32 "Safety, safe driving, fire safety" (ABBREVIATION: "safety")
- 33 "Smoking, alcohol abuse" (ABBREVIATION: "smoking_alcohol_abuse")
- 34 "Domestic violence" (ABBREVIATION: "domestic_violence")
- 35 "Self esteem, bullying, cyber bullying" (ABBREVIATION: "self_esteem")
- 36 "Political candidates (support or opposition)" (ABBREVIATION: "political")
- 37 "Charities" (ABBREVIATION: "charities")
- 38 "Unclear"

Results by classes

Best performing classes

- 0 "Restaurants, cafe, fast food"
- 8 "Cars, automobiles"
- 9 "Electronics"
- 16 "Beauty products and cosmetics"
- 18 "Clothing and accessories"

Poorest performing classes

- 19 "Baby products"
- 36 "Political"
- 35 "Self esteem, bullying, cyber bullying"
- 34 "Domestic violence"
- 33 "Smoking, alcohol abuse"

Results

Correct ad predictions for “Restaurants, cafe, fast food” category:

The image is a composite of two screenshots from a mobile device. On the left, there is a World Vision advertisement featuring a young child with the text "Help save a child's life today" and a "Sponsor a child now" button. On the right, there is a Pizza Hut advertisement with the headline "STOP HUNGER IN ITS TRACKS." and a promotional offer of "\$6.99 EACH 2 FOR \$12.99".

CAUSE
AN EFFECT

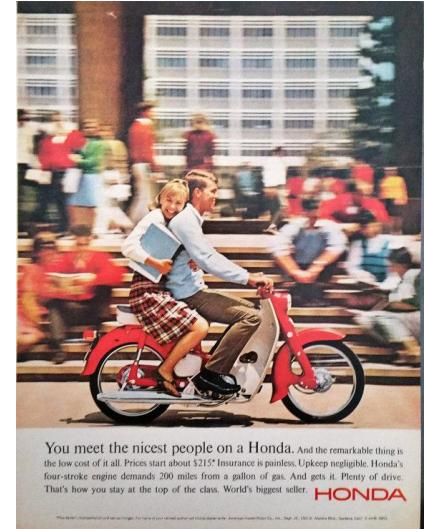


Make dinner a selfless act by joining us for a fundraiser to support U of M Ad Club. Come in to the Chipotle at **800 Washington Ave SE** in Minneapolis on **Tuesday, March 8th** between **1:00pm and 5:00pm**. Bring in this flyer, show it on your smartphone or tell the cashier you're supporting the cause to make sure that 50% of the proceeds will be donated to U of M Ad Club.



Results

Correct ad predictions for “Cars, automobiles” category:



Results

Correct ad predictions for “Electronics” category:

The AOL homepage features a search bar with "Search" and "Homepage Feedback" options. Below the search bar is a "Hot Searches" section with "Christmas train travel" and "Hasselhoff sectioned". The main content includes a "QuickCreditScore" banner, a "Good Excellent Fair Poor" scale, and a "Find Out Now!" button. On the left, there's a sidebar with links to eBay, BBC, Amazon, Wikipedia, AOL Links, Autoblog, Bebo, Bollywood, Cars, Celebrity, Classifieds, Dating, Discount Codes, FanHouse, Film, Flytrap, Games, Homes & Interiors, Horoscopes, Hot Searches, Living, Local NEW, Lottery, Maps, Men, Mobile, and Money. The central part of the page has a large "Intel Core - The heart of Technology" advertisement featuring an Intel Core i7 processor.



Results

Correct ad predictions for “Beauty” category:

Firmer, plumper skin in just 1 week.
THE POWER OF A COSMETIC LIFTING EFFECT IN A SERUM.

RÉNERGIE LIFT MULTI-ACTION
REVIVA CONCENTRATE™
INTENSE SKIN REVITALIZER

NEW
REVITALIZE • REPAIR • REPLUMP

DUE TO HIGH-PERFORMANCE REVITALIZING SERUM
FOR VISIBLE LIFTING, TENSE AND MOIST SKIN.
NEW RÉNERGIE REVIVA CONCENTRATE,
INTENSE SKIN REVITALIZER, ENRICHED WITH AN
ADVANCED PEPTIDE TECHNOLOGY.

YOUR FRIENDS NOTICE IT:
76% OF WOMEN SAY THAT FRIENDS FOUND THE SKIN
LOOKED MORE BEAUTIFUL AND REVITALIZED IN just 2 WEEKS!

WOMEN ARE CONVINCED:
After 1 application everyday, 86% find that their
skin looks revitalized.

VISIBLE RESULTS AFTER 4 WEEKS:
- Skin looks firmer (84%)
- Face contains against fine lines (80%)
- Skin looks denser (92%)

Express your lifting pleasure on lancome.com
To complete your lifting and firming routine,
discover Lancôme Rénergie Lift Multi-Action day, night
and eye creams.

Maybelline Eye Shadow • Eye Liner • Eye Pencil
Maybelline Eye Liner • Eye Pencil

Lancôme

Kate Winslet

Take these 3 easy steps to
INSTANT
Loveliness . . .

Millions of women instantly gain added charm and loveliness with these three delightful, easy-to-use Maybelline preparations. They use *Maybelline Eye Shadow* to accentuate the depth of color of their eyes and to add a subtle, refined note of charming allure. Four colors: Black, Brown, Blue, and Green.

Then—they use *Maybelline Eyelash Darkener* to instantly make their lashes appear dark, long, and beautifully luxuriant—to make their eyes appear larger, more brilliant and bewitchingly inviting. There are two forms of *Maybelline Eyelash Darkener*: Solid form and the waterproof Liquid; either in Black or Brown.

The third and final step is a touch with *Maybelline Eyebrow Pencil* to artistically shape the brows. You will like this pencil. It is the clean, indestructible type, and may be had in Black and Brown.

Take these three easy steps to instant loveliness now. Begin with the Eye Shadow, follow with the Eyelash Darkener, and finish with the Eyebrow Pencil. Then, from the height of your new found beauty, observe with what ease you attained such delightful results. This radiant transformation is achieved only by using genuine Maybelline products. Insist upon them.

Maybelline
EYELASH DARKENER ~ EYE SHADOW ~ EYEBROW PENCIL
Instant Beautifiers for the Eyes

Results

Correct ad predictions for “Clothing and accessories” category:



Results: Incorrect Predictions

True label: "Vacation and travel"

Predicted label: "Clothing and accessories"



True label:
"Restaurants,
cafe, fast food"

Predicted label:
"Financial
services"

Results: Incorrect Predictions

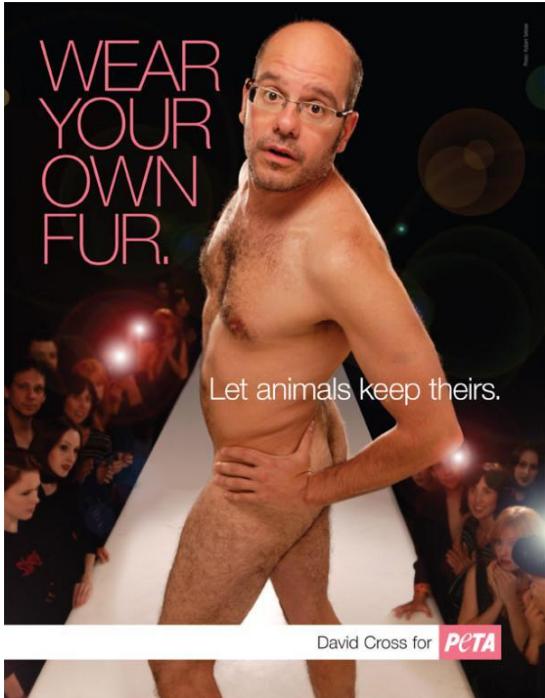
True label: "Chocolate, cookies, candy, ice cream"

Predicted label: "Chips, snacks, nuts, fruit, gum, cereal, yogurt, soups"



Results: Incorrect Predictions

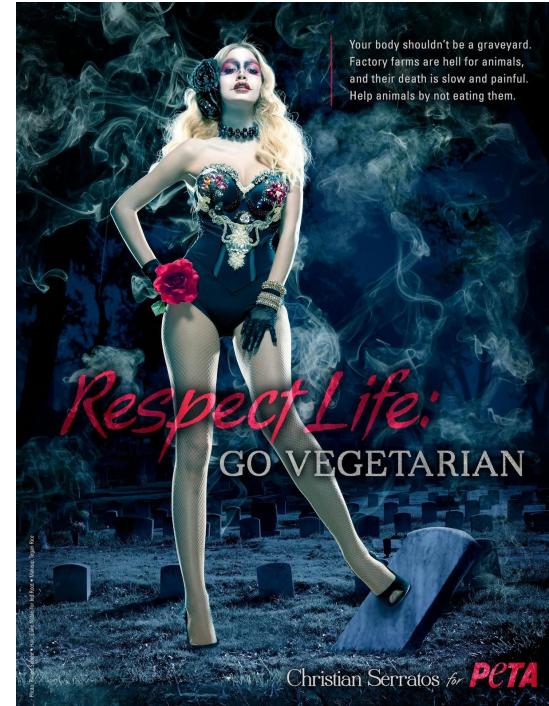
Difficulty interpreting public service announcements (PSA):



True label:
"Animal rights,
animal abuse"

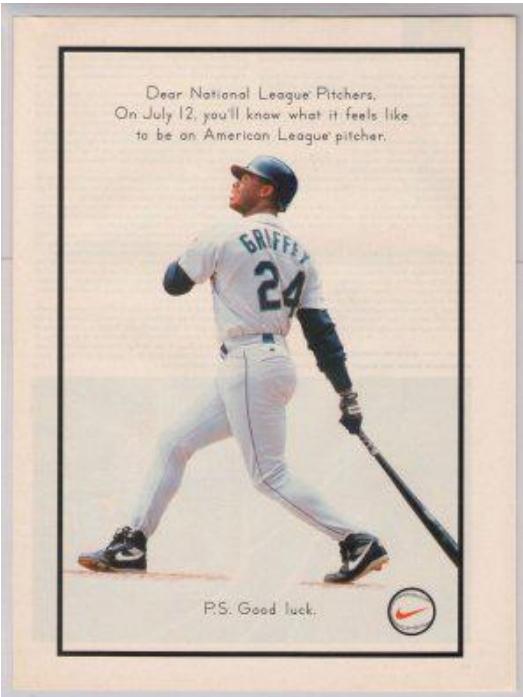
(Left) Predicted
label: "Clothing
and accessories"

(Right) Predicted
label: "Alcohol"



Results: Incorrect Predictions

Many false positives in the "Clothing and accessories" category:



True label:
"Sports
equipment
and
activities"



True label: "Self esteem,
bullying, cyber bullying"

Ways to improve

1. Train on the whole dataset - we used 8.75% of the dataset
2. Try another model architecture
3. Remove “noise” from dataset - annotations aren’t all consistent
4. Add Natural Language Processing component to understand ads with text

Our Code

You can find our code on the github Repo given below.

Github Link : <https://github.com/divya025/Computer-Vision>

Bonus : A little about SGD and Convergence

- ❖ Stochastic gradient descent (SGD) performs a parameter update for *each* training example $x(i)x(i)$ and label $y(i)y(i)$.
- ❖ Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn online.
- ❖ SGD performs frequent updates with a high variance that cause the objective function to fluctuate heavily
- ❖ While batch gradient descent converges to the minimum of the basin the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting.

Any Questions?