OBJECT CLASSIFICATION IN IMAGES VIA DEEP LEARNING WITH OFFERUP

ZHENG HONG TAN, JUNNAN KOU, ZIQIAO XU, SANDEEP J RAMANATHAN
INDUSTRY MENTOR: ALEXANDRA TESTE
FACULTY MENTOR: PROFESSOR RANIA HUSSEIN

INTRODUCTION

> OfferUp

- OfferUp is a mobile based online shopping service provider with 44 million users that allows you to sell everything from clothing to cars.
- They are a C2C marketplace with emphasis on in-person transactions.
- It ranks top 10 of Pacific Northwest private companies (Geekwire 200).

> Project background

- Sellers need to load images of the item and write a verbal description when posting on OfferUp.
- Buyers can find their desired items through search better if OfferUp can show search results based on the images
- Our team can use deep learning and image classification technology to classify images based on type (clothes vs shoes) and category (gender and shoe types).

> Deep Learning

Deep Learning is a type of machine learning that trains a computer to perform human-like tasks, such as recognizing speech, identifying images or making predictions.

> Major steps to build a model

1. Preparing the dataset
2. Training the Model
3. Deploying the model

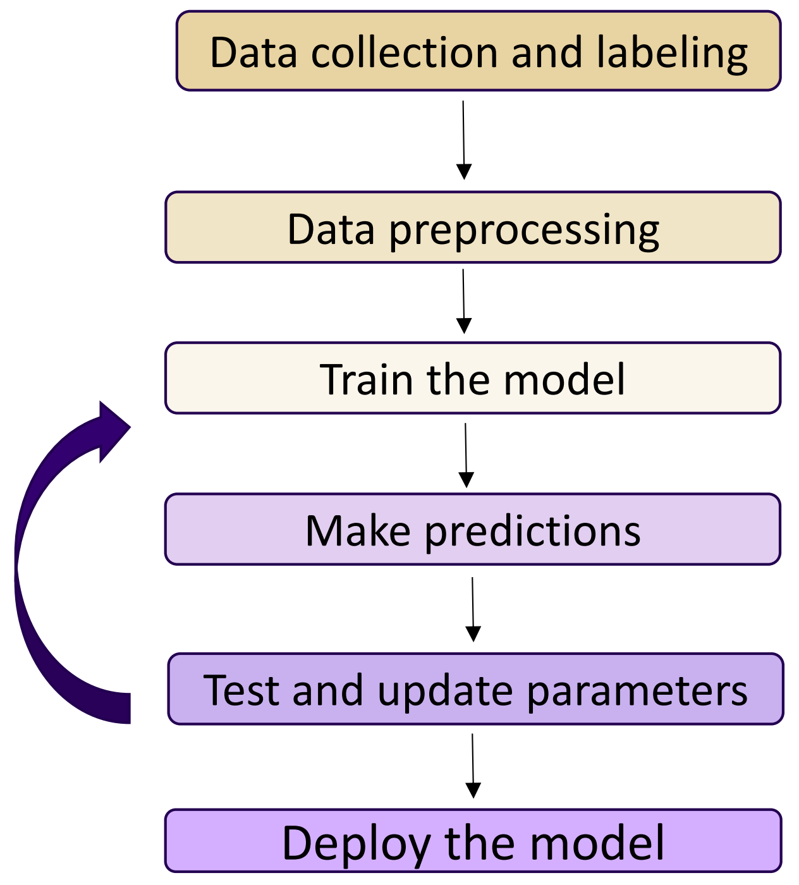

Data collection and labeling
Data preprocessing
Train the model
Make predictions
Test and update parameters
Deploy the model

Figure 1: The entire process of image classification

> Project overview

Model / Technology	Clothes vs. Shoes	Clothes/Shoes gender	Shoe types
Python (RegEx)	✓	✓	✓
Google Cloud Storage	✓	✓	✓
Google AutoML	✓	✓	
TensorFlow 2.0			✓
GCP notebook			✓
Streamlit	✓		
Docker	✓		
XCode	✓		

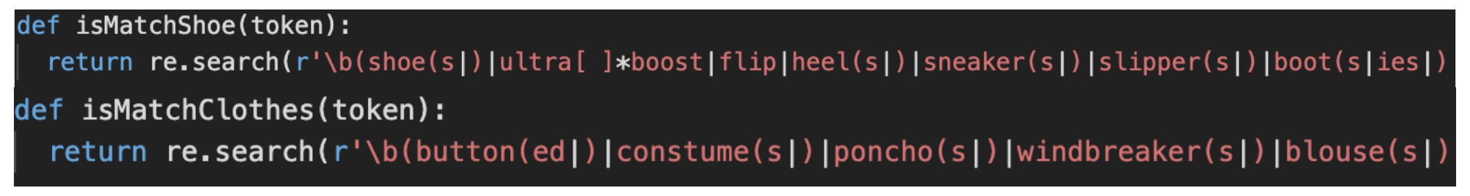
Figure 2: The technologies and resources used in our three image classification models

PREPARING THE DATASET

Generally, preparing the dataset is the most important step in the process of building a model. It is the base on which everything else is built. The clearer your dataset is, the better your model can perform.

> Preliminary Steps

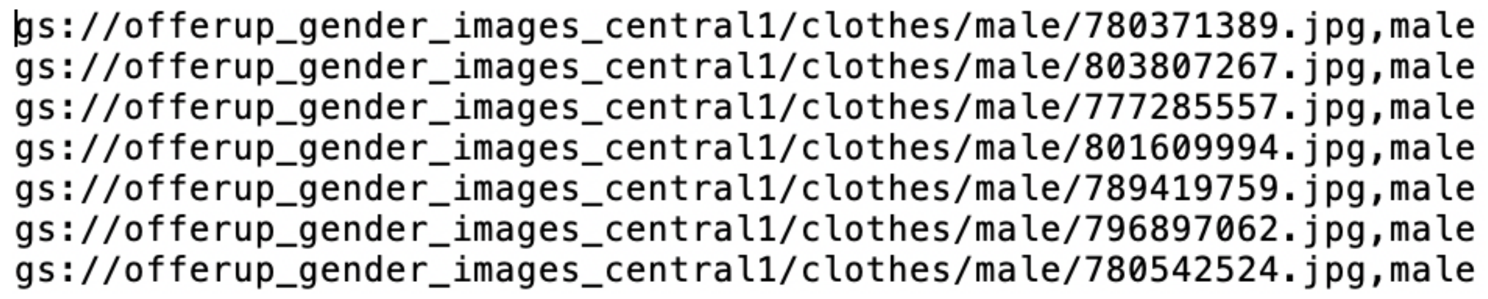
- Download and label images (keyword-based scripts).
- Write scripts that detects regular expressions (RegEx) to create a CSV file to show each image with its corresponding labels.



```
def isMatchShoe(token):
  return re.search(r'\b(shoe(s|)|ultra[ ]*boost|flip|heel(s|)|sneaker(s|)|slipper(s|)|boot(s|ies|)
def isMatchClothes(token):
  return re.search(r'\b(button(ed|)|constume(s|)|poncho(s|)|windbreaker(s|)|blouse(s|)
```

Figure 3: RegEx Python code for shoes and clothes classification

- Organize the images by label.



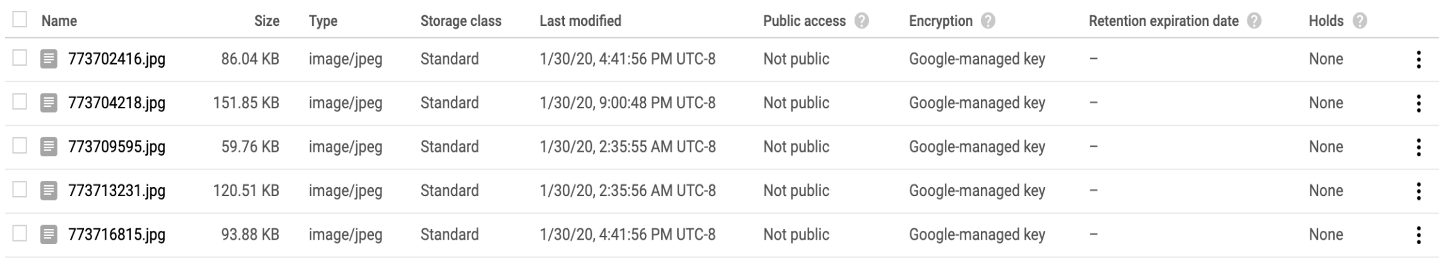
```
gs://offerup_gender_images_central1/clothes/male/780371389.jpg,male
gs://offerup_gender_images_central1/clothes/male/803807267.jpg,male
gs://offerup_gender_images_central1/clothes/male/777285557.jpg,male
gs://offerup_gender_images_central1/clothes/male/801609994.jpg,male
gs://offerup_gender_images_central1/clothes/male/789419759.jpg,male
gs://offerup_gender_images_central1/clothes/male/796897062.jpg,male
gs://offerup_gender_images_central1/clothes/male/780542524.jpg,male
```

Figure 4: CSV file with location to images with their respective labels

> Data/Image preprocessing

Google AutoML

- Create a GCP account and a bucket on GCS.
- Load the original images for AutoML training on their respective category folders.



Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
773702416.jpg	86.04 KB	image/jpeg	Standard	1/30/20, 4:41:56 PM UTC-8	Not public	Google-managed key	–	None
773704218.jpg	151.85 KB	image/jpeg	Standard	1/30/20, 9:00:48 PM UTC-8	Not public	Google-managed key	–	None
773709595.jpg	59.76 KB	image/jpeg	Standard	1/30/20, 2:35:55 AM UTC-8	Not public	Google-managed key	–	None
773713231.jpg	120.51 KB	image/jpeg	Standard	1/30/20, 2:35:56 AM UTC-8	Not public	Google-managed key	–	None
773716815.jpg	93.88 KB	image/jpeg	Standard	1/30/20, 4:41:56 PM UTC-8	Not public	Google-managed key	–	None

Figure 5: GCP image upload

Transfer Learning

- Resize images to 224x224 pixels.
- Perform normalization. Divide the image pixel value by 255 to get a result in the range from 0 to 1.

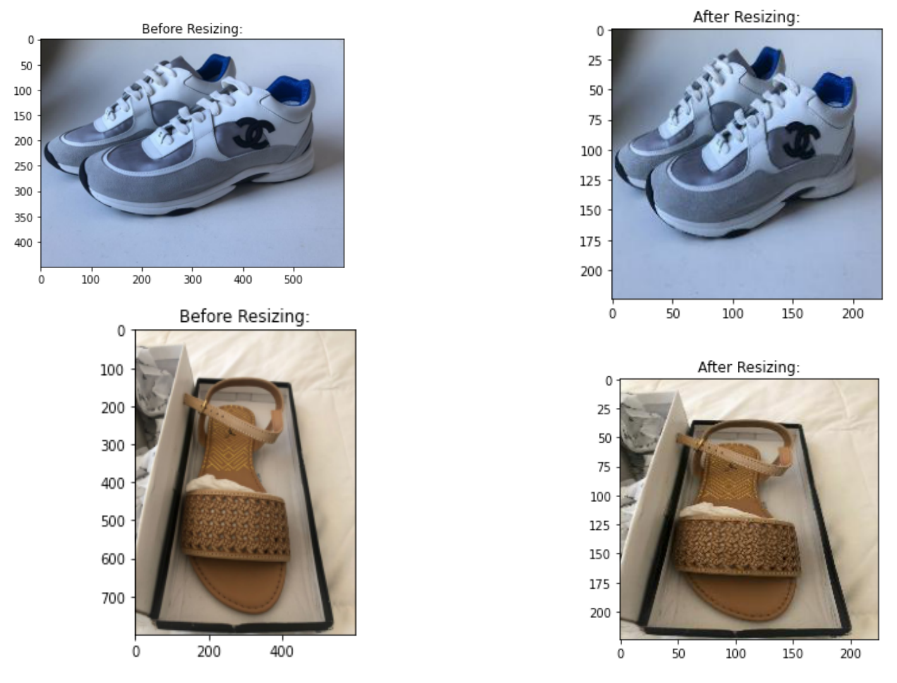


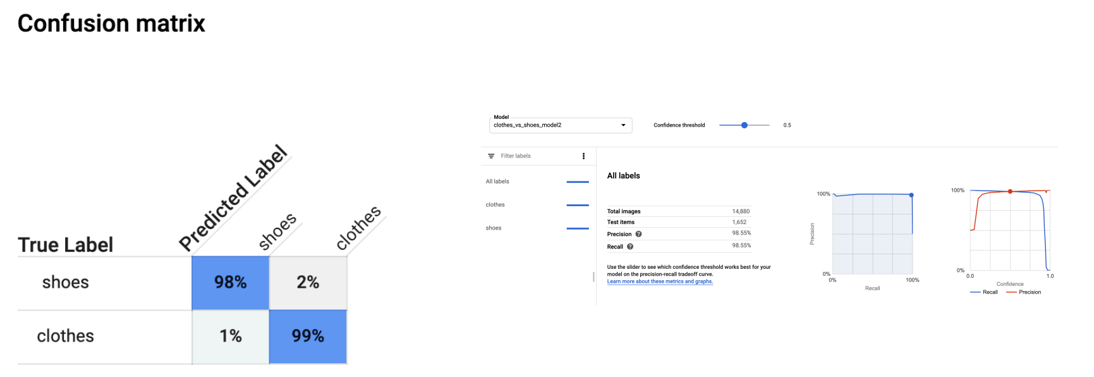
Figure 6: Images before and after resizing to 224x224 pixels

TRAINING THE MODEL

When training the model, one of the metrics that we could use to evaluate how well the model performs on unseen data is the validation accuracy. The better the validation accuracy, the better the model can predict a correct category for which a new item belongs to.

> Google AutoML

- We leveraged Google's AutoML Neural Architecture Search (NAS) to train a model that distinguishes clothes from shoes.



Confusion matrix

True Label \ Predicted Label	shoes	clothes
shoes	98%	2%
clothes	1%	99%

Model: clothes_vs_shoes_model2
Confidence threshold: 0.5

All labels

Total images	14,880
Test items	1,652
Precision	98.55%
Recall	98.55%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
Learn more about these metrics and graphs.

Figure 7: Confusion Matrix and Precision vs Recall Curve of the Clothes/Shoes Model

- We then used AutoML to build a gender model to identify male vs female vs unisex clothes and shoes.
- The gender model did not have excellent accuracy because the product design for some clothes or shoes types like sneakers and boots vary little between genders.

> Transfer Learning

- We decided to classify different shoe types to assist the gender model using Transfer Learning on TensorFlow 2.0.
- We leveraged transfer learning. We tested both VGG16 and MobileNetV2 as the initial architectures. We benefited from this pretrained base model and trained only its last layers along with a few additional layers that we added.

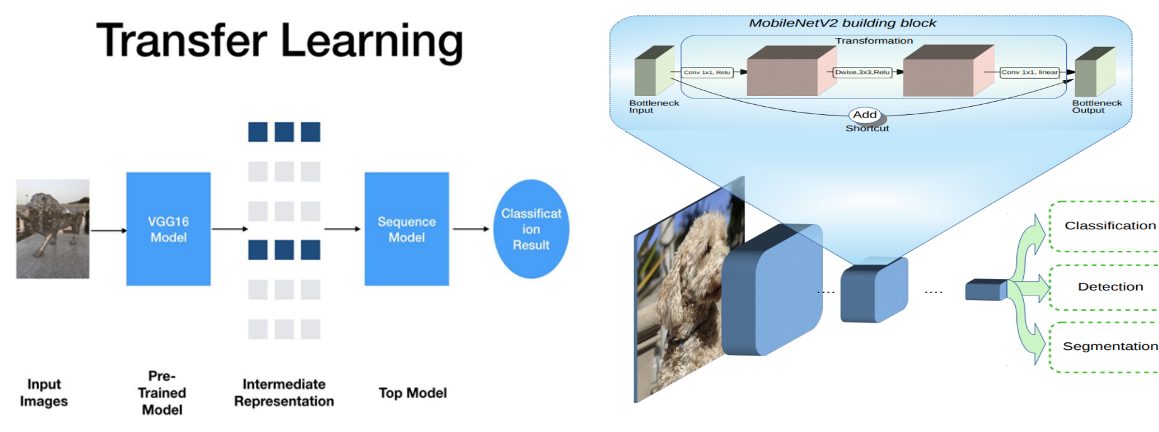
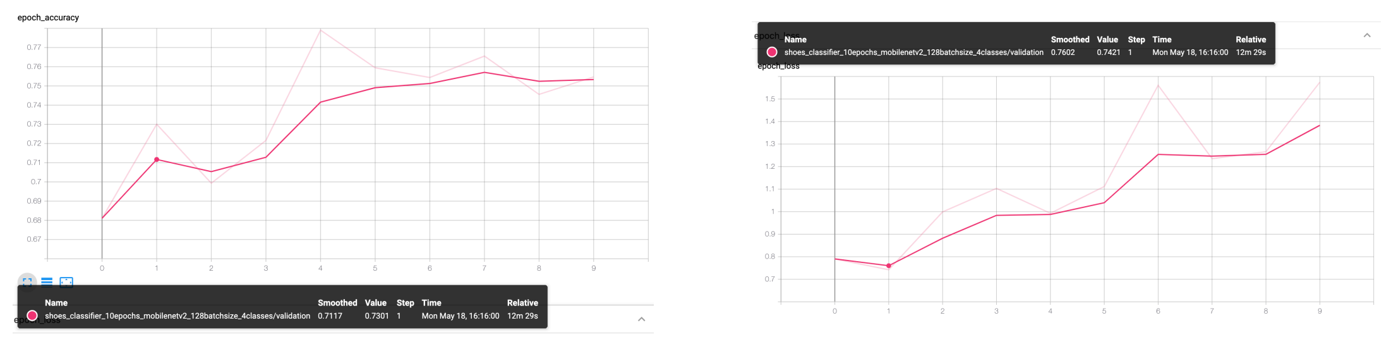


Figure 8: Transfer Learning Architectures of VGG16 (left) and MobileNetV2 (right)

- We tuned the class weights to adjust for class imbalance.
- We evaluated the model's performance and update the parameters in training.



Name	Smoothed	Value	Step	Time	Relative
shoes_classifier_10epochs_mobilenetv2_128batchsize_4classes/validation	0.7117	0.7301	1	Mon May 18, 16:16:00	12m 29s

Name	Smoothed	Value	Step	Time	Relative
shoes_classifier_10epochs_mobilenetv2_128batchsize_4classes/validation	0.7602	0.7421	1	Mon May 18, 16:16:00	12m 29s

Figure 9: Plots of the Validation Accuracy and Loss of the MobileNetV2 Model

DEPLOYING THE MODEL

Once the model is ready for use, there are many ways to deploy it into production. We used a few methods to test out our model.

> GCP Deployment: We first deployed the model onto GCP servers and used the User Interface to upload and test the accuracy of a few images.

> Docker: We then deployed the model into a Docker container and made API calls to the model.

> Streamlit: We then used Streamlit to make an interactive web application where users can upload multiple images and get the predictions.

> iOS and Android apps: Finally we used the TensorFlow Lite (quantized) version of the model to deploy it onto an iOS and android app.

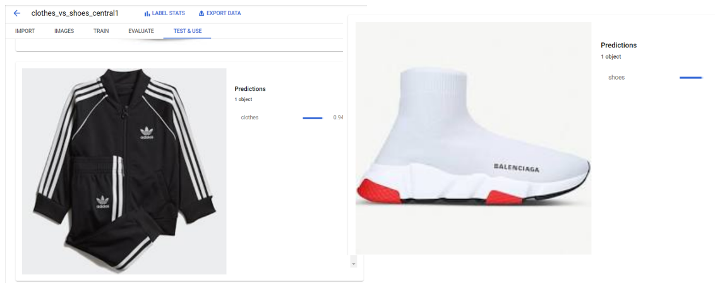
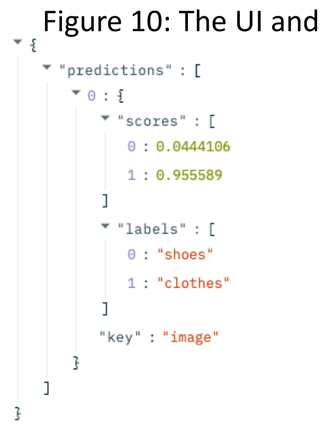


Figure 10: The UI and prediction result on GCP



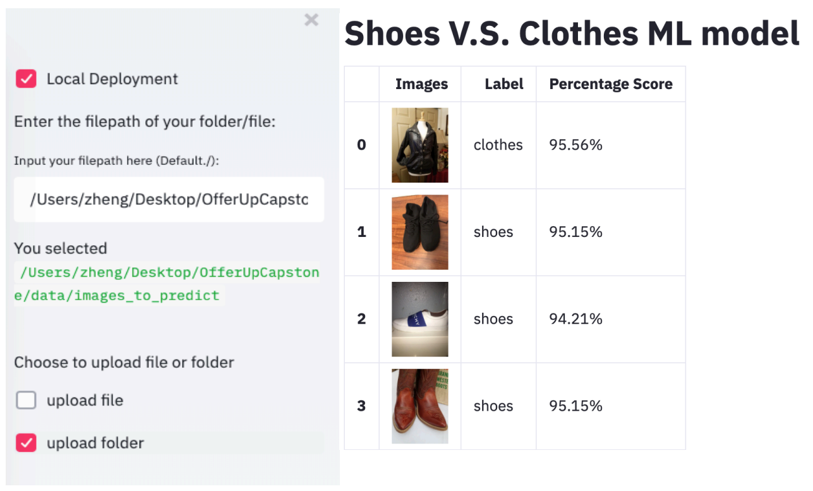
```
{
  "predictions" : [
    0 : {
      "scores" : [
        0 : 0.0444106
        1 : 0.955589
      ]
      "labels" : [
        0 : "shoes"
        1 : "clothes"
      ]
      "key" : "image"
    }
  ]
}
```

Figure 11: JSON Response from Docker container

Prediction: Clothes

Percentage: 95.56%

Figure 12: Prediction Accuracy of Streamlit



☑ Local Deployment

Enter the filepath of your folder/file:

Input your filepath here (Default./):

/Users/zheng/Desktop/OfferUpCapstc

You selected
/Users/zheng/Desktop/OfferUpCapstone/data/images_to_predict

Choose to upload file or folder

☐ upload file

☑ upload folder

Shoes V.S. Clothes ML model

	Images	Label	Percentage Score
0		clothes	95.56%
1		shoes	95.15%
2		shoes	94.21%
3		shoes	95.15%

Figure 13: Streamlit Interface for Users to Upload Images for Prediction

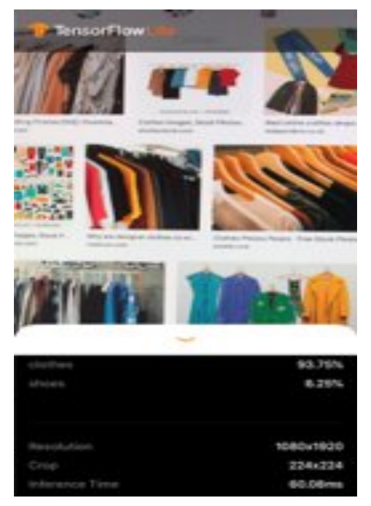


Figure 14: iOS App

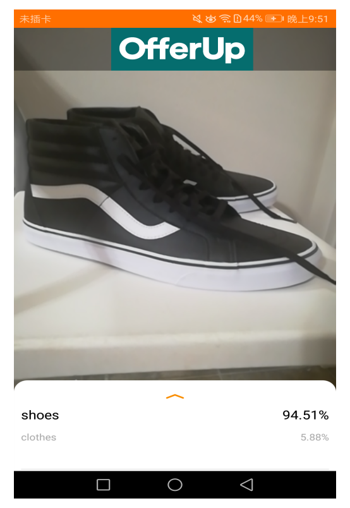


Figure 15: Android App

Future Steps

- Keep improving the performance of the image classification models
- Implement an image search function to find similar images based on the models we trained
- Use the shoes classifier model to label items and build text classifiers.
- Build the image classification completely from scratch instead of using transfer learning.

References

> [1] "Creating Storage Buckets", Google Cloud. [Online]. Available: https://cloud.google.com/storage/docs/creating-buckets>

> [2] "Edge Containers Tutorial", Google Cloud. [Online]. Available: https://cloud.google.com/vision/automl/docs/containers-gcs-tutorial

> [3] peterpetrov826, "Transfer Learning and ImageDataGenerator in Keras," *Kaggle*, 15-Jul-2019. [Online]. Available: https://www.kaggle.com/peterpetrov826/transfer-learning-and-imagedatagenerator-in-keras

> [4] "Transfer learning with a pretrained ConvNet", TensorFlow. [Online]. Available: https://www.tensorflow.org/tutorials/images/transfer_learning