ISyE 6740 - Summer 2023
Project Report

Team Member Names: Vivi Banh (Proposal, Final Report, and Model Evaluation) & Chukwuemeka Okoli (Proposal, Final Report, and Model Development)

Project Title: Deep Neural Network for Shoe Model Identification

1 Problem Statement

In today's fashion-forward society, social media plays a pivotal role in shaping consumer habits and driving trends. As people browse through fashion photos online, they seek convenient ways to identify and purchase the products showcased in those images. However, traditional methods of searching for unknown products on search engines can be frustrating and time-consuming.

Our project aims to address this challenge by developing an image classification system that can accurately identify shoe classes worn by models in photos and provide consumers with the class of the shoe. With an upload of an image, users can quickly get the shoe class, and this potential streamlined approach significantly enhances the shopping experience, making it easier for consumers to find and purchase the specific shoe models they are interested in.

Integrating our image classification system into popular platforms like Instagram or a web application can bring significant benefits to retailers and e-commerce platforms. By seamlessly incorporating a product identification feature, businesses can captivate potential buyers, enhance customer engagement, and drive conversions. This transformative capability bridges the gap between inspiration and purchase, transforming social media platforms into highly effective sales channels. The integration of our image classification system can create new opportunities for both retailers and consumers, maximizing sales potential and delivering an exceptional shopping experience.

2 Data Source

The data utilized for this project is the UT Zappos50K Shoe Dataset (ver 1.2), curated by Yu and Grauman (2014) and available on Kaggle.com. This comprehensive shoe dataset comprises 50,025 catalog images obtained from Zappos.com. The images have dimensions of 136 x 102 pixels, showcasing shoes centered on a white background and consistently oriented for ease of analysis. The dataset encompasses various shoe models. The classes to predict include 20 different class of shoes. We trained a neural network and evaluate the performance of our trained model.

3 Methodology

Our problem is a classification problem since we are trying to determine the category of shoe or class of shoe from an image. of several shoes from different brand names. We utilize a Convolutional Neural Network (CNN) model, a deep learning architecture well-suited for image classification tasks. The CNN model will be trained using the labeled shoe images to learn the distinctive features of each shoe model. We build a custom Convolutional Neural Network (CNN) architecture with three convolutional blocks, followed by a classifier head with two dense layers. The model uses the Keras Sequential API to create the architecture.

3.1 Data Preprocessing

The preprocessing steps we applied including resizing and normalization, to ensure they are in a suitable format for training. Data augmentation techniques such as vertical and horizontal reflections, rotation up to 90 degrees, and vertical and horizontal shifting of the images up to 20% of their original size will be applied to enhance the model's ability to generalize. To preprocess the image, we used the MobileNet V2 preprocessing function from Keras API. To load the images from disk in small batches of size 32, we used an "ImageDataGenerator". We resized the images and used a small images of size 224 * 224. We have 20 classes namely "Ankle

Boots", "Knee High Boots", "Over the Knee Boots", "Prewalker Boots", "Mid-Calf Boots", "Athletic Sandals", "Flat Sandals", "Heel Sandals", "Boat Shoes", "Flats Shoes", "Oxfords Shoes", "Clogs and Mules Shoes", "Firstwalker Shoes", "Sneakers and Athletic Shoes", "Heels Shoes", "Prewalker Shoes", "Crib Shoes", "Loafers Shoes", "Slipper Heels Slippers", "Slipper Flats Slippers", "Boot Slippers". The generator used the folder structure where the images were stored to infer the labels for the images. We split the data into training, validation and testing sets with 70% of the data for training the model, 15% for validating the datasetand selecting the best parameters, and 15% for testing the model.

3.2 Model Architecture

The neural network is organized in layers meaning we take an image of a shoe, pass it through all the layers and get a prediction of what class the shoe belongs too. We used a custom CNN architecture with three convolutional blocks, followed by a classifier head with two dense layers. The convolutional layer consists of many filters giving us multiple feature maps for each filter. The output of one convolutional layer is used as input into the next layer. The structure of the neural network is shown in **Fig. 1** below:

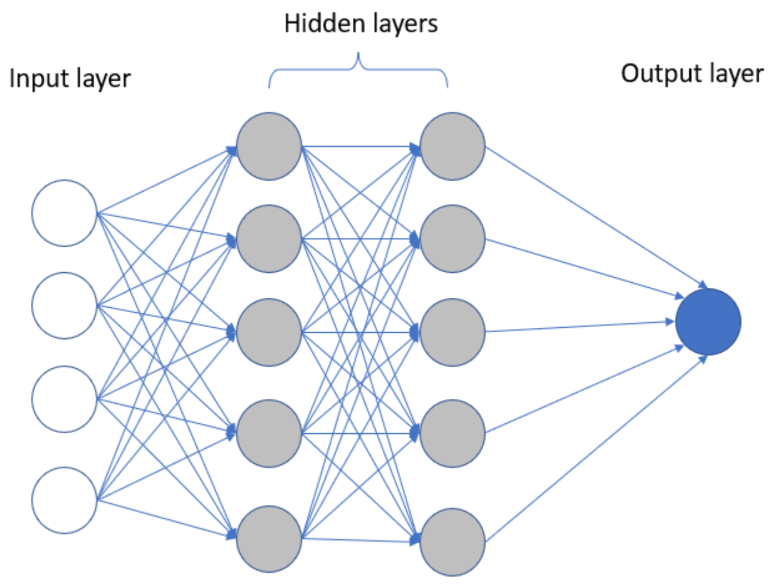

Figure 1: Structure of a neural network

3.3 Model Development

The CNN model is used to train labeled shoe images as input and their corresponding shoe model names as target labels. The model learnt to recognize the visual patterns associated with each shoe model during the training process. To develop the model, we used *Conv2D* as input layer with the first convolutional block followed by a *MaxPooling2D* layer. The MaxPooling2D layer reduces the spatial dimensions of the output by performing max-pooling operation. The second convolutional block is similar to the first one, with 64 filters and a kernel size of 3x3. Again, a MaxPooling2D layer follows the Conv2D layer to reduce spatial dimensions. The third convolutional block is similar to the previous ones, with 128 filters and a kernel size of 3x3. Another MaxPooling2D layer follows the Conv2D layer. After the last MaxPooling2D layer, a Flatten layer is used to convert the 2D feature maps into a 1D vector, which is then fed into the fully connected layers. A Dense layer with 740 units and ReLU activation function is used. Finally, there is another Dense layer with 20 units and a softmax activation function.

This is the output layer, and it represents the probability distribution over the 20 classes in the classification task. The chosen optimizer is “adam,” which is a popular adaptive learning rate optimization algorithm. The loss function used is “categorical_crossentropy,” which is appropriate for multi-class classification problems with one-hot encoded labels. The model is evaluated based on the ”accuracy” metric during training and testing. A schematic of the convolutional neural network developed is shown in **Fig. 2**.

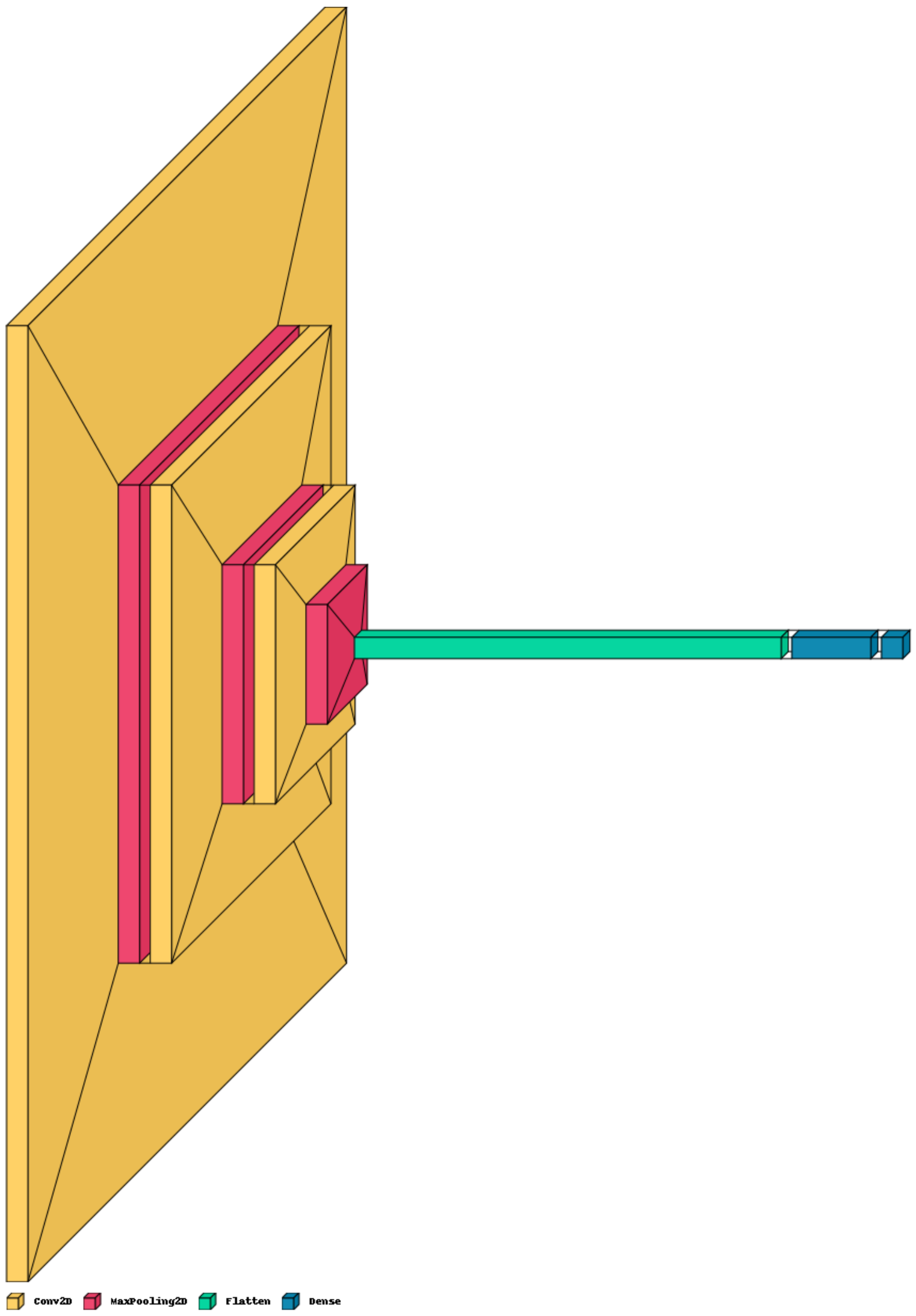

Figure 2: A schematic of the convolutional neural network

In the Adam optimization algorithm, we specify the learning rate, which determines how fast our network learns. We set a reasonable value of 0.01 since if we set it too high, the network learns too fast and skip some details. If it is too low, then the model training time increases significantly.We define the loss function as “CategoricalCrossentropy” which is usually used to train a classification model with multiple classes. We chose accuracy as our metric since we want to determine the percentage of correct classification that the model achieves. We specify the *epoch* which is the number of times the model run through the training data. We set a value of 10 since using more epoch means the model learns the training data better. We don’t want the value to be too high or the model starts to overfit the training data.

4 Evaluation and Final Results

The plot of the loss and accuracy on the dataset is shown in **Fig. 3**. The plot shows there is a large margin between the training accuracy and validation accuracy. The model achieved a training accuracy of 98.84% and a validation accuracy of 82.84%. We can try to increase the overall performance of the model by reducing overfitting as notice by the difference in accuracy between training and validation. To control for overfitting, we use *data augmentation* and add *dropout* to the model. The model augumented the data by generating additional images and transforming the images by flipping, rotating etc. We also applied dropout to the layer. By adding dropout to the model, we set the activation to zero. We choose a dropout value of 0.2 which means 20% of the output units randomly from the layer. By applying data augumentation and dropout, we observed that the model had less overfitting than before and the margin between training and validation error have reduced. A schematic of the convolutional neural network with dropout is shown in **Fig. 4**.

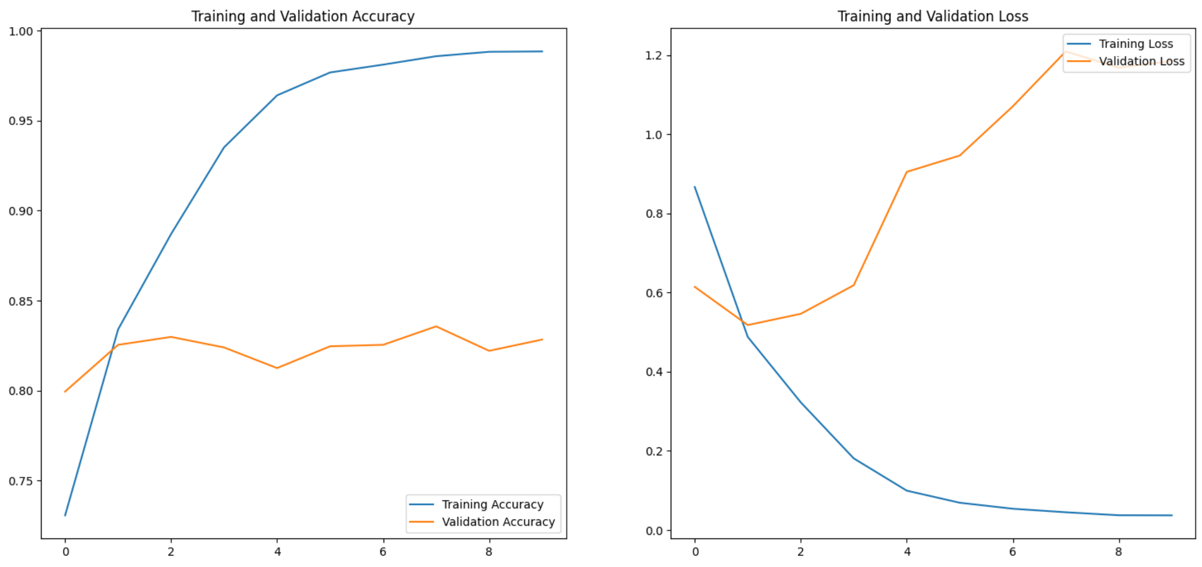

Figure 3: The plot of loss and accuracy on train dataset

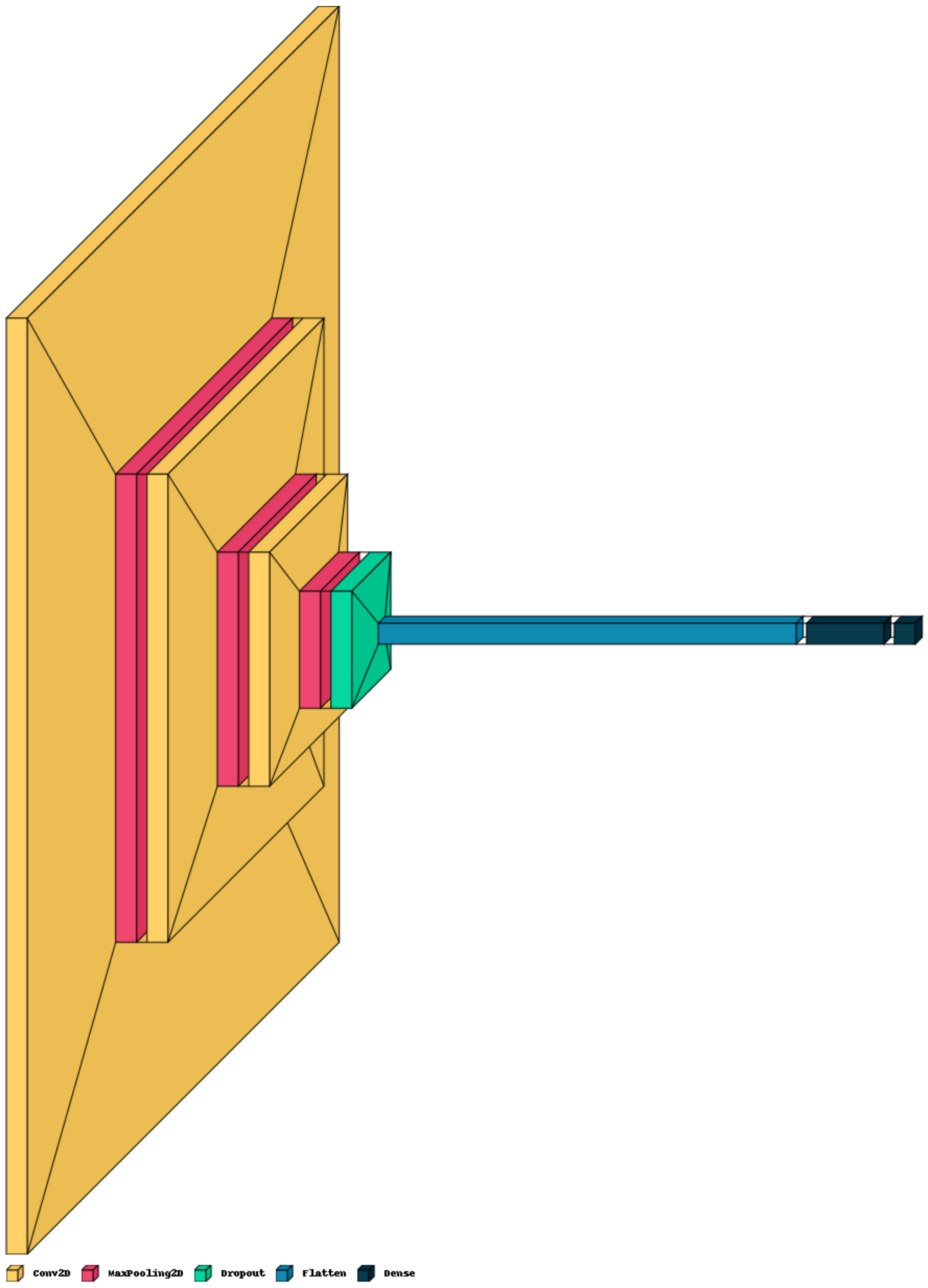

Figure 4: A schematic of the convolutional neural network with dropout

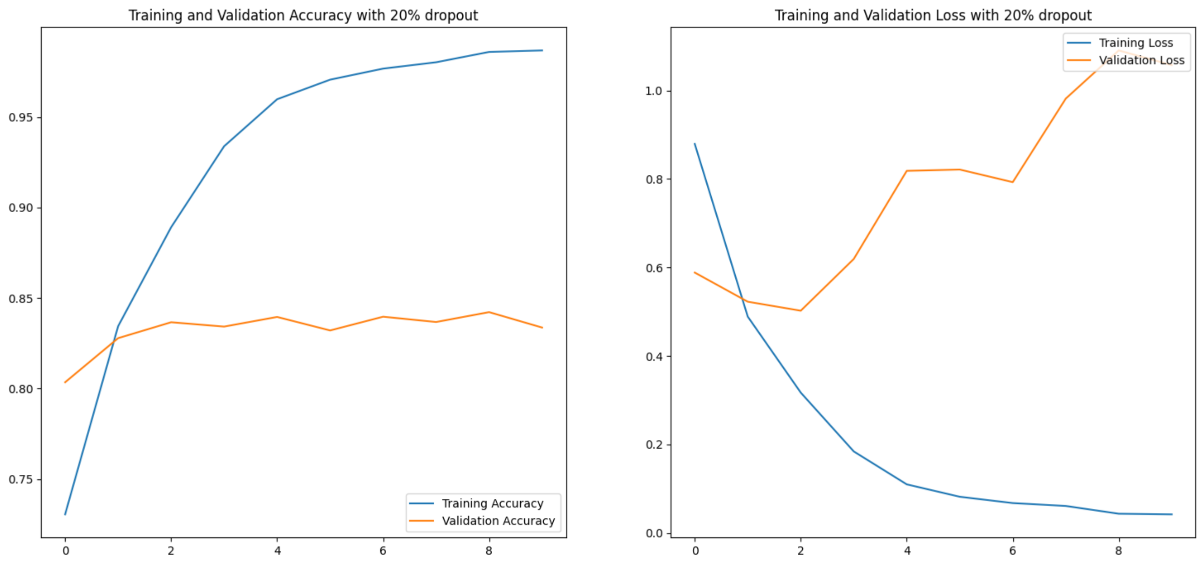

Figure 5: The plot of loss and accuracy on train dataset with dropout

Epoch	1	2	3	4	5	6	7	8	9	10
Loss	0.8665	0.4879	0.3227	0.1810	0.0996	0.0690	0.0539	0.0450	0.0374	0.0371
Train accuracy	0.7307	0.8340	0.8870	0.9352	0.9640	0.9768	0.9811	0.9858	0.9882	0.9884
Validation accuracy	0.7995	0.8254	0.8298	0.8240	0.8125	0.8246	0.8254	0.8357	0.8221	0.8284

Table 1: Result of model training and validation accuracy with epochs

The plot of loss and accuracy on train dataset with dropout is shown in **Fig. 5**. Using our trained model, we then proceeded to make prediction on the test dataset. We used accuracy of prediction as the metric to determine how well our model performed. The accuracy on the test dataset was 81.59% before removing dropout. With 20% dropout removed, we noticed that the model accuracy didn't show any increase at 98.68%. Conversely, the validation accuracy increased to 83.37%. This shows that adding dropout did not leads to improvement in model prediction. Considering that we used a lot of data to train the model, the likelihood of fitting to the noise in the data is minimal at best.

Epoch	1	2	3	4	5	6	7	8	9	10
Loss	0.8791	0.4889	0.3172	0.1843	0.1099	0.0819	0.0676	0.0611	0.0435	0.0421
Train accuracy	0.7305	0.8344	0.8891	0.9339	0.9598	0.9706	0.9767	0.9803	0.9860	0.9868
Validation accuracy	0.8035	0.8278	0.8366	0.8342	0.8395	0.8321	0.8397	0.8368	0.8422	0.8337

Table 2: Result of model training and validation accuracy after dropout with epochs

Using the trained model, we tested the model on the test dataset and perform inference on a sample of images. The prediction on the test set gave an accuracy of 82.7%. The final model developed was a well-trained model capable of accurately identifying shoe worn from shoe image. The success of the project will be determined by the model's ability to achieve a high level of accuracy in identifying the correct shoe models from the images, providing a solid foundation for further advancements in image classification and consumer-driven shopping experiences.

For future enhancements, the trained model can be deployed as a user-facing application to enable a user-friendly experience. Consumers will be able to upload images and receive the corresponding shoe model names as the output. This application will streamline the process of identifying and purchasing specific shoe models, enhancing the shopping experience for consumers. One thing we didn't try out was to compare the performance of our trained model with other pre-trained model like ResNet50 and Fastai models.

5 References

1. Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a Convolutional Neural Network. 2017 International Conference on Engineering and Technology (ICET), 1–6. https://doi.org/10.1109/icengtechnol.2017.8308186

2. Sharma, N., Jain, V., & Mishra, A. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. Procedia Computer Science, 132, 377–384. https://doi.org/10.1016/j.procs.2018.0

3. Yu, A., & Grauman, K. (2014). Fine-Grained Visual Comparisons with Local Learning. Computer Vision and Pattern Recognition. https://doi.org/10.1109/cvpr.2014.32