CSCI 1430 Final Project Report: Multi-label Classification on PASCAL VOC 2007 Dataset

Team Zero: Changcheng Fu
Brown University

Abstract

Unlike traditional classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or "labels." This paper introduce a pipeline solving multi-label classification on PASCAL VOC 2007 dataset. It implements three different models including self CNN models, VGG-16, and ResNet50 and performs comparative experimental results of certain multi-label classification methods.

1. Introduction

Nowadays, machine learning becomes a good technique for us to solve more social issues in a wiser way. While involving in large-scale data, such as music, news, and pictures, it's helpful for people to gain an efficient method to create labels for those large data in order to classify their features. Multi-label classification involves predicting zero or more class labels.

Multi-label classification involves predicting zero or more class labels. Unlike traditional classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or "labels." As a result, it raise my attention on design pipeline for solving multi-label classification tasks with large data set.

Multi-label classification is a more difficult task than single-label classification because both the input images and output label spaces are more complex. Furthermore, collecting clean multi-label annotations is more difficult to scale-up than single-label annotations. In this paper, we utilize the deep neural network to carefully making PASCAL VOC 2007 dataset into logical inputs like one-hot vector for each classes and then link them with images input. Then to learn with prepossessed labels and images, we introduce a new classifier created by our own with sigmoid activation function and binary cross-entropy as loss function. Our approach allows the use of the same training settings as when learning with all the annotations. We further explore transfer learning with more deeper models like VGG-16 and ResNet50 which already pre-trained in ImageNet to hopefully increase our accuracy to make better prediction.

If we can successfully build the pipeline and reach a relatively good result with our model. It's helpful for us to label text, sound and video files, etc. Especially, those elements may simultaneously belong to several topics and in result have multiple tags/labels. Rather than just giving yes or no answers like single-classification task, it can now categorize music, news, photos efficiently.

2. Related Work

Keras model.[2] Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. It contains plenty of implementations of commonly used neural-network building blocks like layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. In our project, we fully utlized its ability as implementing our own model architectures and utilized the pre-defined ResNet50 model and did modification efficiently.

ResNet.[3] Deep convolutional neural networks have a series of breakthoughts in past few years. Kaiming and his team in Microsoft Research defines Resnet to utilize the residual learning companying with shortcut connections in their design to make their network no only deeper but remains details from previous layers accurately.

3. Method

In this multi-label classification task, we face two main problems: data processing and classifier training. For data processing, we need to utilize the PASCAL VOC 2007 and convert the necessary information in data for multi-label classification task. In addition, all images need to link with annotations and classes file in order to make useful inputs as data generator so that our model can learn with logic. Then for training classifiers, we need to make classification and prediction for images and let model can be consistently defined for each categories and output multiple features for images if possible.

3.1. Pipeline

We create a pipeline(Fig. 1) to solve the multi-label Classification problem by several steps:

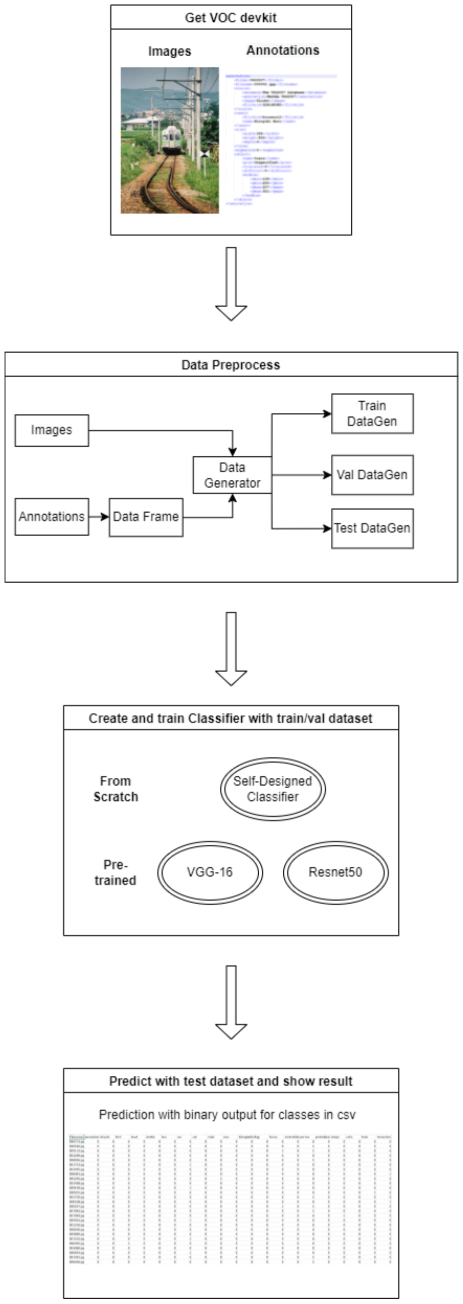


Figure 1. Pipeline for project.

We download official open resource VOC devkit from website and separate it out as Images(.jpg), Classes(.txt) and Annotations(.xml). By using classes and annotations, we can link the specific class for each image filename in order to create data frame for each images and then create data generator by linking image path with preprocessed data frame. Then, we implement and compile three different types of model(one from scratch, two loading from pre-trained weights and trained the last convolutional layer along with fully connected layers after) using Keras, and then fit models with our data generators. After 100 epochs finding the best weights by evaluation, we use the checkpoint with highest accuracy to predict the test generator and output the value to csv for visualization. Finally, we link the csv file with image by using the filename to perform multi-label classification to images. In addition, we compare the output with ground truth label from generator with the output from csv file to see whether our prediction is correct visually.

3.2. Data

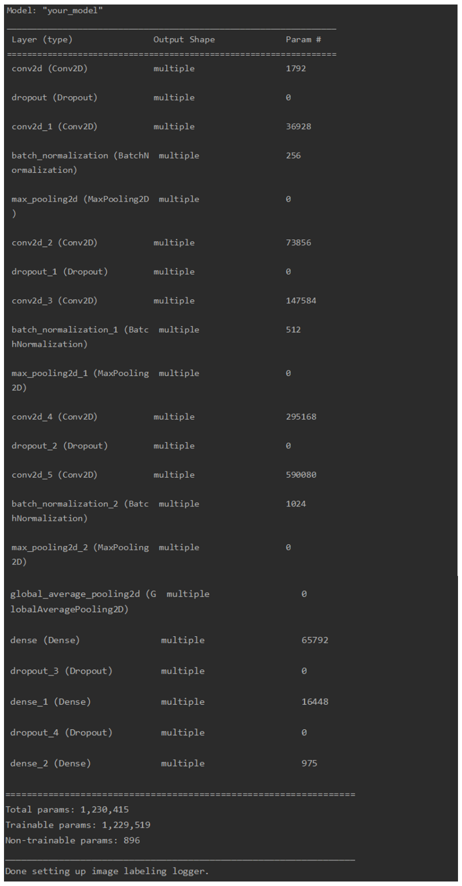
Data becomes a main part for this project and we utilize the classes file VOC data set provided for us which contains the information that link filename with the specific classes. Then, we iterate all the data file in order to make a column list.

```
for item in VOC_CLASSES:
    f = open(data_path + item + dataset
        , 'r')
    item_list = f.read().splitlines()
    f.close()
    item_list = [e for e in item_list
        if ('-' not in e)]
    column_list = []
    for e in item_list:
        e = e.split(" ", 1)[0]
        column_list.append(e)
    allColumn_lists.append(column_list)
```

3.3. Model Architecture

We have implemented and tested various plain/residual nets. To provided instances for discussion, we used one self-designed model along with two pre-trained model as follows for PASCAL VOC 2007 data set.

Own Model. Our plain own model(Fig. 2) without loading pre-trained weights is mainly inspired by the philosophy of VGG nets.[5]



```
Model: "your_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             multiple                  1792

 dropout (Dropout)           multiple                  0

 conv2d_1 (Conv2D)           multiple                  36928

 batch_normalization (BatchN  multiple                 256
 ormalization)

 max_pooling2d (MaxPooling2D  multiple                 0
 )

 conv2d_2 (Conv2D)           multiple                  73856

 dropout_1 (Dropout)         multiple                  0

 conv2d_3 (Conv2D)           multiple                  147584

 batch_normalization_1 (Batc  multiple                 512
 hNormalization)

 max_pooling2d_1 (MaxPooling  multiple                 0
 2D)

 conv2d_4 (Conv2D)           multiple                  295168

 dropout_2 (Dropout)         multiple                  0

 conv2d_5 (Conv2D)           multiple                  590080

 batch_normalization_2 (Batc  multiple                 1024
 hNormalization)

 max_pooling2d_2 (MaxPooling  multiple                 0
 2D)

 global_average_pooling2d (G  multiple                 0
 lobalAveragePooling2D)

 dense (Dense)               multiple                  65792

 dropout_3 (Dropout)         multiple                  0

 dense_1 (Dense)             multiple                  16448

 dropout_4 (Dropout)         multiple                  0

 dense_2 (Dense)             multiple                  975

=================================================================
Total params: 1,230,415
Trainable params: 1,229,519
Non-trainable params: 896
_________________________________________________________________
Done setting up image labeling logger.
```

Figure 2. Own model architecture.

It used two convolutional layers with batch normalization, relu activation function, and maxpooling as a block. Batch Normalization is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). This effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift".[4] In addition, we utilize max pooling to extract the sharpest featrues from image with less amount of computational resource and use average pooling to calculates the average value for patches of a feature map, and uses it to create a down sampled (pooled) feature in order to be more efficient.

VGG nets.[5]For using VGG, the convolutional layers mostly have 33 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. VGG gives us an insight that increasing depth using an architecture with very small (3×3) convolution filters can shows a significant improvement on the prior large convolution filters with more deeper layers.

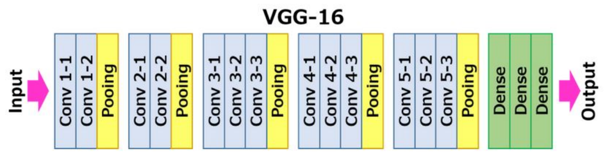


Figure 3. VGG model architecture.

Its architecture accepts input as a fixed-size $224 \times 224 \times 3$ (RGB) image. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3×3(smallest size to capture the notion of left/right, up/down, center) and follow with max-pooling over a 2×2 pixel window, with stride 2. (Fig. 3)

Different from previous model which always contains batch normalization, VGG finds that normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. More importantly, rather than using relatively large receptive fields in the first convolutional layers (e.g. 11×11 with stride 4), they accumulated small convolutional layer to reach the equal receptive field.

ResNet.[3] In ResNet(Fig. 4), the basic structures stemmed from VGG nets with the same small 3×3 filters design. In addition, they perform down sampling directly by the convolutional layers with stride of 2 and using global average layer when the colutional layers end, which ResNet has fewer filters and lower complexity compared to VGG nets: 34- layer baseline has 3.6 billion FLOPs (multiply-adds), which is only 18% of VGG-19 (19.6 billion FLOPs).[3]. Other than basic CNN layers, ResNet utilize Deep Residual Learning as the strategies for its model.

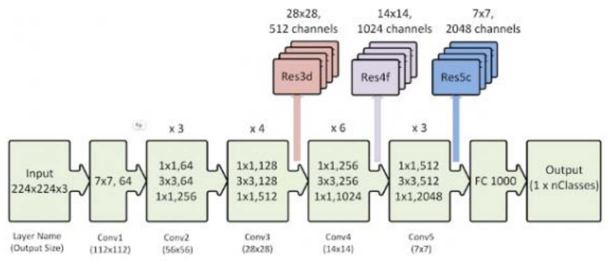


Figure 4. ResNet model architecture.

3.3.1 Residual Learning

They make an assumption based on multiple nonlinear layers can asymptotically appromiate the residual functions. In specific, they can let stacked layers to approximation a residual function $F(x) := H(x) - x$ to reach the outcome $F(x) + x$. For pratical implementation, if the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart, which vanish the degradation problem that previous deep neural network will have. If identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings.

3.3.2 Shortcuts for identity mapping

By using the theoretical residual learning strategies, they create two bottleneck architectures (Fig. 5) which the first uses in shallow layers and the second uses in deeper layers with 1 x 1 convolutions, where the 1 x 1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3 x 3 layer a bottleneck with smaller input/output dimensions.

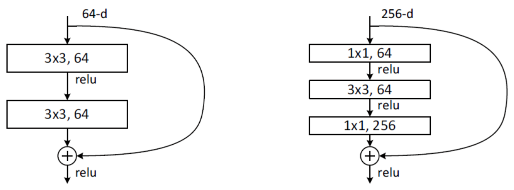


Figure 5. Bottleneck architecture.

4. Results

After finishing the training process and applied your model to the validation data set for evaluation. Our own model reach the accuracy of 46.35% at epoch 69. After that, the pre-trained VGG-16 model with our fine-tuned last convolution layer along with dense layers reach the accuracy of 53.12% at 88 epoch. Finally, the ResNet50 reach the highest accuracy of 69.79% at epoch 63.(Fig. 6)

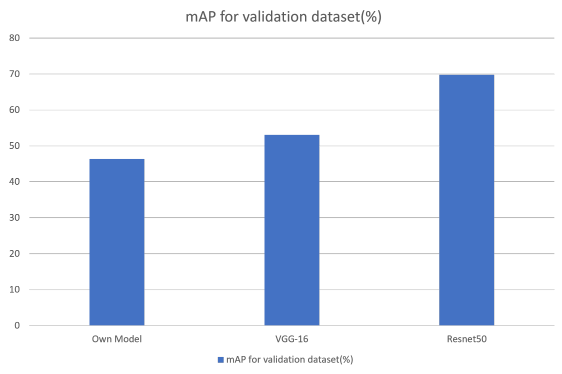


Figure 6. Accuracy for different models

There are pictures and data showing the training in details.

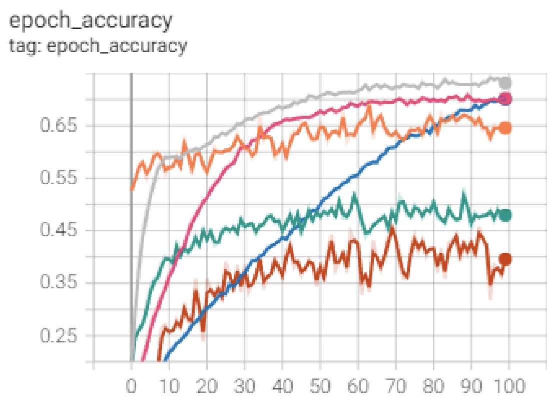


Figure 7. Accuracy change for different models with epoch during training

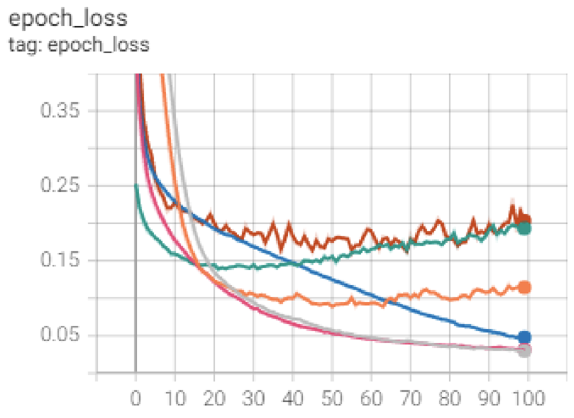


Figure 8. Loss change for different models with epoch during training

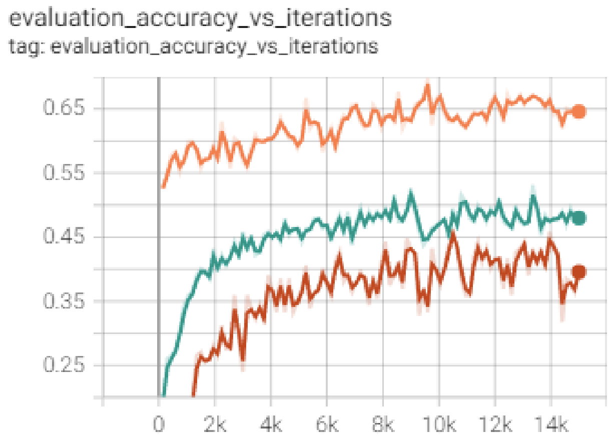


Figure 9. Accuracy change for different models with epoch during evaluation

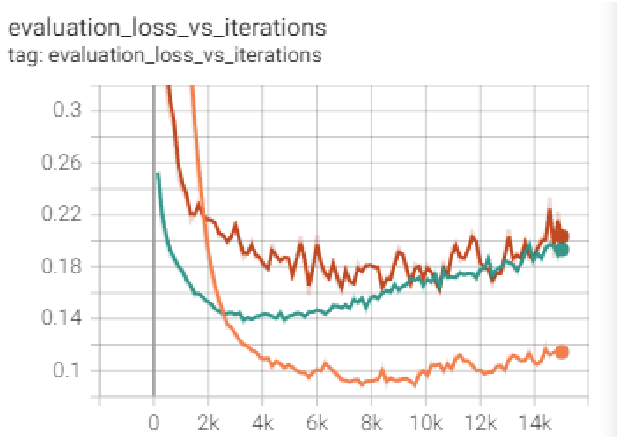


Figure 10. Loss change for different models with epoch during evaluation

4.1. Technical Discussion

What about your method raises interesting questions? Are there any trade-offs? What is the right way to think about the changes that you made?

Multi-label classification requires different metrics than those used in traditional single-label classification. So we use different methods compared to original single-label task. Like using sigmoid function instead of softmax since all the classes are not mutually exclusive since they can appear same time for prediction.

Further, we think that it's impossible or us to notify the difficulty among different layers. There are certainly layers like "cow", "sheep", and "dog" which looks similar to each others inside pictures so that when we using layers like max pooling or average pooling, the features cannot classify them well. More importantly, when we give an input image with all the objects getting close to each other, it's hard to extract features for a narrow space inside the image.

As a result, although we utilize our optimizer and loss function with model to reach a relatively good accuracy on PASCAL VOC data set, there are still limitations for specific layers and limitations for our model to gain important data from pictures.

4.2. Societal Discussion

Cited from the paper about the roadmap of multi-label classification[6], we know that based on the social value, data sets are not getting equally multi-label. During some similar scenes, the number of labels of specific classes are always larger than others. For example, when classifying the image with tvmonitor, there are always more chairs and tables near the tvmonitor. This could be a parameter that influences the performance of the different multi-label methods. Please respond to the following questions. Different projects will have different scales and qualities of impact; we ask you to think creatively and consider the broader implications of your project rather than just the more narrow current technical capability. Responses should together take up roughly one page in your final report.

In addition to that, Another group proposed the trending challenges [1] in multi-label classification that High dimensionality of label space in multi-label data sets. In multi-label classification most labels are associated with a few number of training instances in comparison to the total number of instances in the data set. This situation is similar to the problem of imbalance class distribution in single-label classification. And the situation will be more worse when the number of labels in the data set is very high. Compared to this societal system, we need a simple but fast algorithm to handle large number of labels which associated with few number of instances.

For the field of multi-label classification, a successful model can help companies in advertisement, ranking engine, and etc. to categorize large amount of input they gained with specific labels. It's help for them to gain a stable and high-accuracy pipeline to treat big-data with a initial classification like that. However, the improvement of classification methods will decrease the job opportunities on those experienced workers who always make labels for input images.

In the future work, based on further development on multi-label classification, we need to spend more time doing data augmentation before using as input in order to eliminate those correlation between different classes and localize with more accurate features.

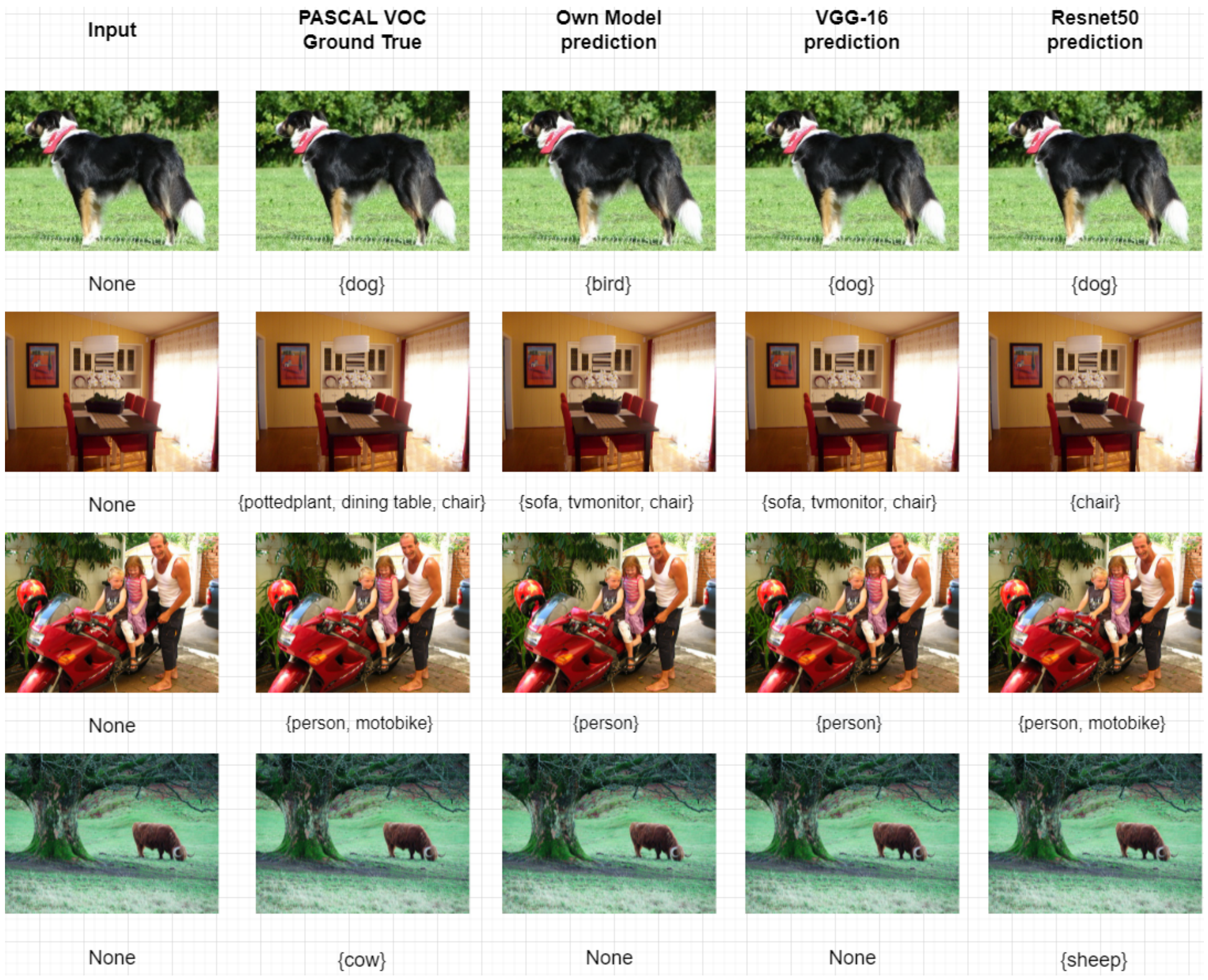


Figure 11. Prediction with different models for random test images

5. Conclusion

In this paper, we have introduce a pipeline to solve multi-label classification task on PASCAL VOC 2007 data set by using different classifiers. Also, we conducts the differences between each classifiers in order to accurately recognize what improvements those model architectures have compared to previous deep neural networks. The main contribution of this paper is providing a solution to predict images with multi-label as shown above(Fig. 11). In the near future, we aim to further increase our model accuracy by creating more advanced models along with more balanced data set.

References

[1] Raed Alazaidah and Farzana Kabir Ahmad. Trending challenges in multi label classification. *International Journal of Advanced Computer Science and Applications*, 7, 10 2016. 5

[2] Francois Chollet et al. Keras, 2015. 1

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1, 3

[4] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?, 2019. 3

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. 2, 3

[6] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 09 2009. 5

Appendix

Team contributions

Please describe in one paragraph per team member what each of you contributed to the project.

Changcheng Fu Since it's a solo project, he designed the pipeline for the whole project. Finish the data preprocessing along with the design and implementation for model. After training for several epochs, he makes analysis towards outcome with graph and table. In addition, he takes charge of the poster creation and writes the final report for the group.

Acknowledgement

We would like to thank some open resource and tutorials for inspirations.

https://github.com/thatbrguy/Multilabel-Classification

https://vijayabhaskar96.medium.com/tutorial-on-keras-flow-from-dataframe-1fd4493d237c

https://github.com/Kenneth-ca/holbertonschool-machine_learning/tree/master/supervised_learning/0x09-transfer_learning