Multilabel Image Classification on VOC 2012 DataSet

Anonymous Author(s)
Affiliation
Address
email

Abstract

In this paper we take on a problem of multilabel image classification of 5 classes with limited amount of data (VOC 2012) and computing resources. Another common challenge we face in the realm of multilable classification is the lable tail problem (imbalanced dataset). We introduce a simple to solve the problem. Lastly, we compare the effectiveness of the result of skills used like data augmentation, hyperparameter tuning, creating cnn with 720 million parameters and using pretrained models.

1 Introduction

From 2015 deep learning has regained popularity with the significant increase in computing power and the development of GPUs by the gaming industry. Moreover, in the era of cloud computing these resources are widely available to anyone with a computer and a strong internet connection.

With such improvements, significants improvements in the field of computer vision has also been inevitable. With deep networks like AlexNet, GoogleNet, ResNet, and Xception, there has been significant decline in the error rate of classifying objects in an image.

One branch of image classification is multilabel image classification. The challenge is harder and more complex than the simple one i.e multiclass image classification. The latter requires detecting a single label that is associated with the input image. However, in the case of multilabel image classification, the input image may have more than one objects and the model must be able to predict the presence/absence of the objects.

2 Literature Review

2.1 Mutli-Label Image Classification by Feature Attention Network [1]

The paper introduces two of the most common problem present in multilabel image classification. The label tail problem where some classes are underrepresented and the object scale inconsistencies where some objects comprises of only few pixels compared to other objects. The paper then talks about promising ways how to deal with such inherent problems related to the multiimage classification realm. The first method talks about Feature Attention Network that harnesses informative, fine grained features and unique features of underrepresented classes and obtain high accuracy. Moreover, with the help of correlation learning network that uses attention mechanism (input interact with each other to identify who should they pay more attention to) for learning label correlation, semantic and sparital dependencies among the features.

Submitted to 34th Conference on Neural Information Processing Systems (NeurIPS 2020). Do not distribute.

By using such information the network is able to achieve high accuracy. The model is tested on MSCOCO 2014 and Pascal VOC 2007 dataset to demonstrate the effectiveness and versatality of the solution

2.2 C-Tran for multilabel image classification [2]

The models uses transformers to exploit the dependencies and relationship of the labels with each other. In the training phase the model randomly hides some labels, and uses the given input image and the unhidden labels to predict the hidden labels. This way the model learns to use the surrounding (given) labels in the images as well. In the testing phase, with the extra labels provided the models attempts to predict the rest of the labels. As it is least likely for a cow to be present with a whale in a given image, or it is highly likely for a skyscraper to be present in the same image of sky, the model relies strongly on using label correlation for predicting the presence and absence of labels.

The model is able to handle labels with extra labels, or partial labels or no labels at all. The durablity of the model in handling different types of inputs is a novelty of the paper. The method is called Label Mask Training where it allows the model to generalize to any label settings.

Moreover, the models performs better than models like ResNet101 or CNN-RNN and is able to achieve state of the art in all the given datasets (COCO and Visual Genome)

3 Pre Experimentaion and challenges

The following picture depicts the statistics of number of classes present in the VOC 2012 dataset (borrowed from the VOC website)

The result and the methodology of the art are described in the following sections

4 Methodology

The data are heavily imbalanced. For example, there are 1994 images of person present in the training data. However, about one tenth of images pertains to the animal "cow". We test this imbalance data using a CNN built from scratch with about 21 million parameters with dataset comprimising on person, aeroplane, bicycle, car, cat and car. We see a accuracy of 70 percent in traning. However, when evaluating the model on a test data comprimising of images containing only the person, the accuracy reaches near 100 percent. Interestingly, the accuracy barely touches 10 percent when the test images contains everything but person.

This explains an important concept in the field of machine learning. The model has learnt the features of humans very well as the input data is heavily skewed towards person. Therefore, it performs exceptionally well in recognizing person given an image. However, since the number of images for other classes are comparatively low than the person class, the model fails to capture the unique features (patterns) of the other four classes. Reading several articles, I realized class imbalance is a common problem in multilabel image classification. Articles [3] demonstrates some promising ways to tackle the issue by using different performance metric and using data augmentation.

However, since the project requires only 5 classes in the multilabel problem, I decided to choose those 5 classes that are about the same in numbers. I also made sure to pick the classes that are high in number. This way the model can have as many data to be trained with. These classes were bird(395),car(590), cat(539), chair(566), and dog(632). Running the model with the same architecture, caused the accuracy score to go down to 10 percent during training. As a first instinct I suspected the model with 21 million parameters might not be too simple for a complex data. Therefore, I bumped the parameters to about 768 million parameters by decreasing the strides in all layers, and significantly increase the neurons in the dense layers. My attempt to intentionally overfit the data failed as the accuracy score remained almost the same. I also employed intense hyperparameter tuning like chaning the activation function from relu to tanh, selu or using a different weight initialization like "he-initializer" (as i suspected the expoding/vanishing gradient problem might be the issue). However, the accuracy score did not show any significant improvements. Using data augmentaion might have made the problem more worse, as the model stuggled to fit with the current data.

I suspected the data might be still unbalanced as the number of images pertaining to the five chosen classes comprimised of about 2500 images while the remaining half of the training images consisted of images not containing any of the chosen five classes. In order to fix the heavy imbalance, I performed downsampling of the data by incorporating only 260 (almost equal to the frequency of bird class) images that did not consist any of the chosen five classes. The change resulted in a 20 percent overall accuracy score. (Note: I decided to stick with the accuracy metric instead of precison, recall or f1-score as the frequency of the five classes were almost the same (balanced dataset). After doing the necessary preprocessing and gaining a mere 20 percent in the accuracy, I suspected the CNN created from ground up might be flawed in harvesting the essential features of the input images.

Therefore, I decided to rely on pretrained models and did some study on transfer learning[4].

Many pretrained models were studied to choose the most ideal pretrained model. Upon research Xception model was selected for the problem. Firstly, Xception consisted of the inception module used by GoogleNet. The inception module enabled GoogleNet to make efficient use of parameters. Moreover, it had 10 times fewer parameters than AlexNet. The inception module has four different branch. Each branch with different sizes of filters captures patterns at different scales. Moreover, the module also consist of 1X1 kernels. Even though these pixels cannot detect features as they study only pixel at a time, they can capture patterns in the depth dimension of the feature maps. Using these unit size kernels also reduces number of parameters. Contrary to a linear fashioned CNN, model incorporating the inception module is able to run different pairs of filters across the input capturing more complex patterns.

Taking the inspiration of the inception module by GoogleNet, Xception module also uses residual connections of the ResNet. In a typical CNN, when the lower layers have not started learning, the higher layers are also not able to run. Instead of feeding the output of inner layer to the input of the next higher layers, skip connections allows the output of inner layers to feed into the input of other higher layers. This allows the higher layers to start making progress if the lower layers are stuck, thereby increasing the convergence speed.

The Xception module combines best of both the worlds and takes the inception module to the extreme level (getting the name Xception). Instead of applying the regular inception module, Xception replaces the inception module with depthwise seperable convolution layer. The depthwise seperable convolution layer is different from the standard filter that simulatneously captures spatial patterns (circles, oval) and cross-channel patterns (eyes, mouth, ears). Instead of trying to capture diffrent types of features all at once, depthwise convolutional layers aims to capture spatial and cross channel patterns at different levels. Xception module first applies a standard filter to the feature maps, and the second layers (depthwise convolution layer) looks for cross channel pattersn only. Similar to the inception module, the second layer has filters of 1X1 size, captuaring patterns depthwise. As stated in [4], "depthwise seperable convolutional layers use fewer parameters, less memory, and fewer computations than regular convolutional layers, and in general they even perform better, so you should consider using them by default", Xception model with slight modifications in the upper layers was used for the multilabel image classification

The Xception model has the typical operations of applying convolution operations with relu activation functions followed by batch normalizations to prevent overfitting and seperable convolutional layers to study the cross-channel patterns of the image and the standard max pooling operations. Similar process is repeated multiple times making the model 71 layers deep and with 22.8 million parameters.

We download the pretrained model pretrained on imagenet images. As the imagenet consist of all the five classes we are trying to classify, we hope to achieve good performance. The Xception network is a CNN, it expects images of size 224X224 and rgb pixel values (0-255). We achieve the requirements by using the preprocessing function of the Xception model. As the Xception model originally built for multiclass classifications we had to slight modifications to the structure. We removed the output layer that incorporates softmax function and replaced it with dense layers with 5 neurons and the appropriate sigmoid activation function as the classes are independent of each other. Our modified model has 20,871,725 parameters. As the model are pretrained and the added layer stills needs training, we freeze the pretrained layers and only train the higher layer. After the training phase, we then unfreeze the remaining layers and set a lower learning rate of 0.01, to prevent damaging the already trained layers. We then put our model to test using the test set (images that our model has never seen before).

As we are dealing with sigmoid function, where the classes are independent of each other, we use binary cross entropy as our loss function.

5 Experimentation

5.1 Dataset Description

	train		val		trainval	
	Images	Objects	Images	Objects	Images	Objects
Aeroplane	327	432	343	433	670	865
Bicycle	268	353	284	358	552	711
Bird	395	560	370	559	765	1119
Boat	260	426	248	424	508	850
Bottle	365	629	341	630	706	1259
Bus	213	292	208	301	421	593
Car	590	1013	571	1004	1161	2017
Cat	539	605	541	612	1080	1217
Chair	566	1178	553	1176	1119	2354
Cow	151	290	152	298	303	588
Diningtable	269	304	269	305	538	609
Dog	632	756	654	759	1286	1515
Horse	237	350	245	360	482	710
Motorbike	265	357	261	356	526	713
Person	1994	4194	2093	4372	4087	8566
Pottedplant	269	484	258	489	527	973
Sheep	171	400	154	413	325	813
Sofa	257	281	250	285	507	566
Train	273	313	271	315	544	628
Tvmonitor	290	392	285	392	575	784
Total	5717	13609	5823	13841	11540	27450

As we can see from the table above [5], the dataset consist of 20 classes. The training data consist of 5717 images and the validation set has 5823 images. As described earlier the dataset is imbalanced, with the person class being present 1994 and 2093 in training and validation data respectively. Classes with the lowest frequency include animals like cow and sheep and are present in about 160 images in both the training and validation set. This makes sense as most of the 19 objects are likely being used by a human. For example, in a picture of bicycle, there is likely a person sitting on it. Or there are high chances that someone is sitting on a boat or holding a bottle.

Since the VOC 2012 data contains a dedicated folder for image classification challenge ("Main" Folder), the preprocessing steps and loading the images were decently simple.

As the data is significantly low to be trained with a Convolutional Neural Network heavy reliance are made on data augmentation and network pretrained on millions of similar images.

5.2 Description of the performance measure

Accuracy was chosen as the performance metric [5]. As a refresher, the accuracy is the sum of true positve and true negative divided by the sum of sum of all cases (True Positive + True Negative + False Positive + False Negative). In the case of multilabel classification, keras calculates the correct predictions (True Positive and True Negative) individually. For example, if the predicted label is [0,1,0] and the actual lable is [0,0,0], then the accuracy would be 2/3 = 0.66 (2 True Negatives/All the cases).

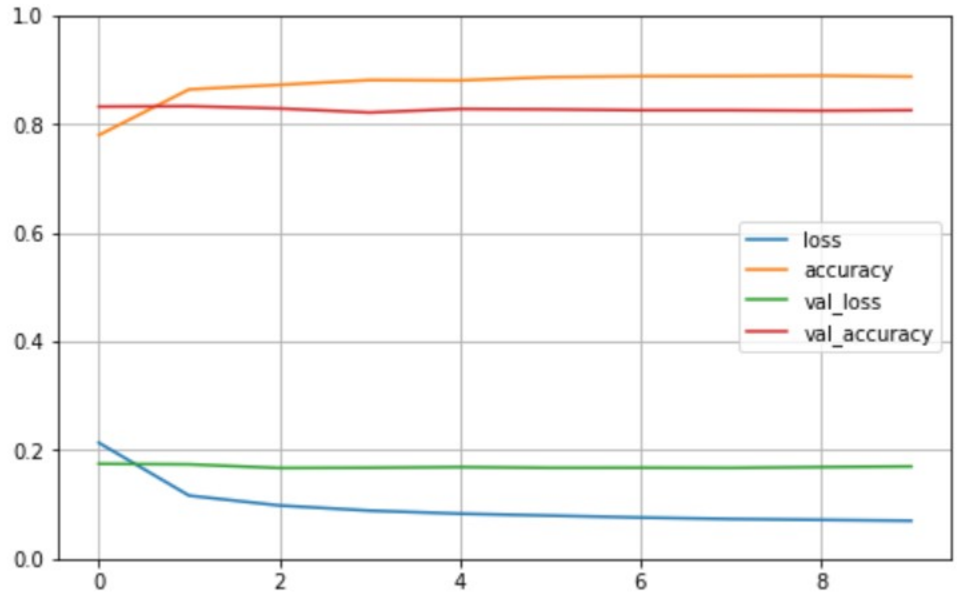
If the dataset would have been imbalanced then the accuracy metric would have been a misleading result.

For example if the y:pred = [1,0,0,0,0,0,1] and the y=[0,0,0,0,0,1,1], the True Negatives = 4, False Negative = 1, True Positive = 1, False Positive = 1, then the accuracy would be (true negative + true positive) / (all the cases) = 5/7 = 0.71. The result is surprisingly high as predicted the presence of only one class (label =1) correctly. Moreover, if there is alot of images with the ground truth label of y=[0,0,0,0,0,1,1] (an imbalanced dataset), and a model that keeps predicting the same ypred: [1,0,0,0,0,0,1], then the accuracy will be continue to be high.

However, since the data is balanced, accuracy would yield a correct estimate in our case. If our data was imbalanced then we would have to rely on precision and recall score that also took into account of other cases ignored by accuracy.

5.3 Validation Process

The training data consisted of 2968 images and the validation data consisted of 2969 images. Since the validation data was almost equal to the training data, I divided the validation data into validation and test data. Having a test data is crucial as our model has never seen the data and represents the real test data, giving us an accurate measure of the generalization error. The validation consisted of 1200 images and the test data consisted of the remaining 1769 images. It is crucial to have validation so we can make sure that our model is not overfitting. Here is graphical representation of training accuracy score, accuracy loss, validation accuracy score and validation loss.



As we can see from the graph, the trianing accuracy starts almost at 80 percent in the first epoch and gradually goes beyond 90 percent at the end of training. The validation accuracy denotes how the model performs on a data that it has not trained on. The validation accuracy remains around 83 percent which is acceptably close to the training accuracy, signifying our modified Xception model is performingly well and not overfitting the data. We also shuffle the training data after every epoch so the model does not see the training data in the same order at every epoch. This ensures our model is immune to bias in the data.

We see a similar trend when it comes to the training and validation loss. The training loss signifies the amount of error of the model in the training set while the validation loss signifies the error of the model on the validation set. As both the loss are close to each other, we determine our architecture is not overfitting.

We finally run our model in the test set and achieve an evaluation score of 86 percent which is quite impressive.

6 Discussion and conclusion

Obtaining an accuracy of 10 percent initially to achieving a score of 86 percent might seem like a clear win. However, the solution classifies only 5 labels given an image and is far from the state of the art models like C-Tran. However, we gain some interesting and insighful information tackling the multilabel image classification challenge. Firstly, relying on pretrained models when training data is small is an effective strategy. The jump in the accuracy from 10 percent to 86 percent, confirms our strength of pretrained networks like the Xception. when data is significantly low using a pretrained model is very effective.

As a next step, I will try to do some research on state of art promising techniques like employing label correlations and harnessing fine grained unique features by studying models like the C-Tran or the Feature Attention Network.

Moreover, since my experimentation on hyperparemeter was severly limited by lack of GPU (due to poor internet connection), I will try to gain access to the free GPUs offered by Google Colaboratory for my future projects.

7 References

[1] Zheng Yan, Weiwei Liu, Shiping wen, and Yin Yang: Mutli-Label Image Classification by Feature Attention Network

[2] Jack Lanchantin, Tianlu Wang, Vicente Ordonez, Yanjun Qi, University of Virginia: Genreal Mutlilabel Image Classification with Transformers

[3] Aurelien Geron Hands On Machine Learnign with Scikit-Learn, Keras and Tensorflow