Sound generation for Images based on environment recognition

Kaustubh Raval
MS in Computer Science
California State University, Sacramento, CA, United States
kaustubhraval@csus.edu

Raghavendra Sreenivas Acharya
MS in Computer Science
California State University, Sacramento, CA, United States
raghavendrasreeniva@csus.edu

ABSTRACT

Scene recognition is one of the key tasks of computer vision that enables the definition of the significance of object recognition. With the availability of large datasets like ImageNet, SUN and Places dataset and the rise of Convolutional Neural Networks (CNNs) for detecting high-level features, performance at scene recognition has not attained the same level of epitome. Here, we propose new methods to identify the environment of which the image belongs to and then play the audio regarding the predicted environment. Using CNN, we learn deep features for scene recognition tasks and set up new, state-of-the-art results for a number of Scene-centric datasets. A visualization of the CNN layers' responses enables us to see the differences in the internal representations of object-centric and scene-centric networks. In this paper, we explore the use of machine learning algorithm for predicting the scene or environment and play the audio of that particular environment.

INTRODUCTION

One of the major features of our Human body that astonishes us, is the way eye and brain work. It works so efficiently that the scene we see is processed simultaneously and the brain recognizes what category of environment it belongs to. The structure of the human eye is very complex. It functions in a very similar way to a camera, with an opening through which the light enters, a lens for focusing and a light-sensitive membrane at the back. The retina is covered in millions of light-sensitive receptors called as rods and cones. Each receptor contains pigment molecules, which change shape when they are hit by light, triggering an electrical message that travels to the brain via the optic nerve. MIT neuroscientists claim that the human brain processes the image which the eye sees in less than 13 milliseconds. This rises the thought, when machines will be capable of detecting the scene/environment from images just like the human brain.

Nowadays machines are capable of functioning like human brain. Neural Networks is a set of algorithms that help the machine in recognizing the deep relations between the data like how the brain processes it. Here, we try to teach the machine on the basic scenes or environments which are as below:

- Beach
- Building
- Desert
- Forest
- Sky
- Lake

Furthermore, the top two predictions are taken and the audio for the top most prediction will be played.

PROJECT FORMULATION

This section defines more on the problem statement and approach taken to get things straight. The problem being predicting environment from the image by machines and the solution being Convolutional Neural Networks (CNNs).

2.1 Dataset

What does it take to reach human-level performance with a machine-learning algorithm? In the case of supervised learning, the problem is two-fold. First, the algorithm must be suitable for the task, such as Convolutional Neural Networks in the large scale visual recognition and Recursive Neural Networks for natural language processing. Second, it must have access to a training dataset of appropriate coverage and density.

The first task was to finalize the list of scenes or environments we are considering for training and testing the model. Once we have a list of scenes, the next task is to collect images belonging to each scene category. We are considering two datasets here in our project.

I. Sun

The SUN database is the apt dataset with a large coverage of natural scenes with objects. Furthermore, it provides us the unprecedented opportunity to obtain the natural statistics for objects and scenes, and study their relationship.

This dataset is referred as "SUN" (Scene UNderstanding) database. SUN database provides researchers in computer vision, human perception, cognition and neuroscience, machine learning and data mining, computer graphics, and robotics, with a comprehensive collection of annotated images covering a large variety of environmental scenes, places and the objects within. To build the core of the dataset, we counted all the entries that corresponded to names of scenes, places, and environments, using WordNet English dictionary.

For each class of scene, photos are collected from different search engines on the internet using WordNet. Only colour images of 200 × 200 pixels or larger were selected so as to maintain quality. Each image was examined to confirm whether or not it fit a detailed, verbal definition for its category. For similar scene categories (e.g. “abbey”, “church”, and “cathedral”) explicit rules were formed to avoid overlapping definitions. the final dataset reaches 899 categories and 130,519 images.

Here, we are using SUN2012 database, which is created scene centric when compared to other dataset like ImageNet (As shown in Fig 1**[3]**). SUN2012 has around 16,873 images and is approximately of a size 8GB. We considered approximately 15 categories from the dataset and merged it into the 6 final categories.

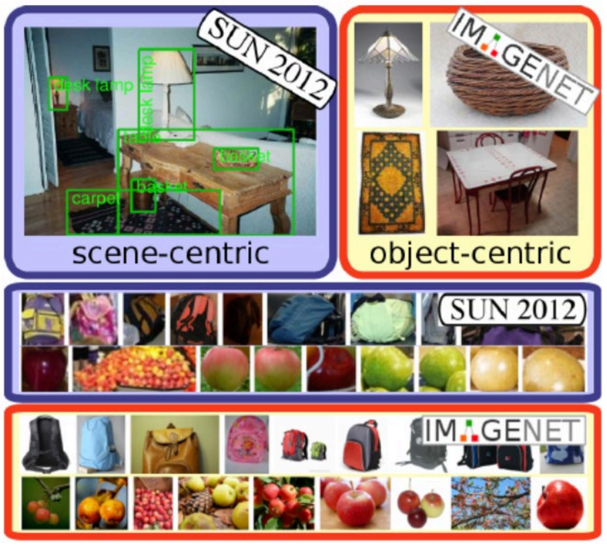

Figure 1. Dataset bias of ImageNet and the SUN2012 database

II. Places365

Places database is also a scene centric database. The strategy of Places is to provide an exhaustive list of the categories of environments encountered in the world, bounded by spaces where a human body would fit (e.g. closet, shower). The SUN (Scene UNderstanding) dataset provided that initial list of semantic categories. The SUN dataset was built around a quasi-exhaustive list of scene categories with different functionalities, namely categories with unique identities in discourse. Places offer a wide variety of images and categories than SUN. A dataset to be good, it has to be dense and diverse. For example, if the dataset has hundreds of images of the same building, the dataset will be dense, but not diverse. Here, Places is built considering these factors and it is much more dense and diverse than SUN. The places dataset comes in different options as follows:

Places365-Standard has 1,803,460 learning images ranging from 3,068 to 5,000 with the number of images per group. There are 50 images per class in the validation set and 900 images per class in the test set.

Places365-Challenge contains the same categories as Places365-Standard, but the training set is significantly larger with a total of 8 million training images. The validation set and testing set are the same as the Places365-Standard.

Places205 has 2.5 million images from 205 scene categories. The image number per category varies from 5,000 to 15,000. The training set has 2,448,873 total images, with 100 images per category for the validation set and 200 images per category for the test set.

Places88 has 88 common scene categories among the ImageNet, SUN and Places205 databases. Places88 contains very less images when compared to other places dataset.

As we are briefing to working on 6 categories of environment we will have to find the respective sounds for these environments and also enough data or images for the model to have a good fit. Hence we will be merging data/images from both the datasets.

Selected Dataset

In our project, we have 6 different categories named beach, building, forest, lake, sky, desert and in each of the category we have 430 images each.

2.2 Data Preprocessing

For a computer, an image is a two-dimensional array of numbers (also called pixels) ranging between 0 and 255. It is defined by the mathematical function f(x,y) where x and y are the two co-ordinates horizontally and vertically. The value of f(x,y) at any point is giving the pixel value at that point of an image.

Firstly, the dataset is categorized into the above mentioned 6 categories and they are structured in different folders for easy identification. This dataset is then fed into our program, with os.path function, we will scan through the folders for the given path and convert them into a list of categories. In our case, it will generate the list of [beach, building, desert, forest, lak e, sky].

Secondly, two new lists are created named all_labels[ ] and all_images[ ] which will be inputted with labels and images to the respective lists. With the help of these two lists, we are creating a dictionary. The dictionary will point the respective labels to the images. Furthermore, to increase efficiency we are using pickle function which will dump the dictionary to a pickle object file. This will help us to load the data directly from the pickle.obj and we wont have to do all of the previous processes again. I will also help save memory. The list from the fetched pickle object is converted to a numpy array using np.array.

Thirdly, the images are resized to a resolution of 256, 256, 3 using a powerful image processing algorithm called OpenCV which has inbuilt resize function. The labels are the same values throughout the dataset, it is better to encode them, hence they are passed to a one-hot encoder which creates a binary column for each category and returns a sparse matrix or dense array. Pandas library is used to perform one-hot encoding. The image is now resized and now it needs to be normalized. To normalize the array, we are dividing the value with 255 which is the maximum value so that we get the values in between 0 and 1. The data is now pre-processed, and the array now contains normalized RGB values. This data is passed to the model.

Lastly, After all the pre-processing, we are generating two pickle object files one for the label and the other for the images and dumping the data into it so that we can directly fetch the data from the generated pickle file. The data is then splitted into X_train, Y_train, X_test and Y_test using SkLearn with the test size of 20% and random state value set to 42. This will give us an X_train_shape of (2158, 256, 256, 3) and Y_train_shape (2158, 6).

2.3 Algorithm Used : Convolutional Neural Networks

Deep Learning is a very similar to the process of human and brain functioning the way the brain learns things. A Convolutionary Neural Network (ConvNet / CNN) is a Deep Learning algorithm capable of capturing an input image, assigning significance (learnable weights and biases) to different aspects or objects in the image and being able to distinguish between them. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics. The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. The images are passed to the CNN which goes through certain layers.

The Layers of Convolutional Neural Network Includes:

1. **Convolutional Layer**

The first layer on CNN is always a Convolutional Layer. The input to this layer is a 256 x 256 x 3 array of normalized numbers/pixel values. Now we select a small area called a filter. For example, take an image with a bird in it, the beak of the bird can be a small area, so we can take that as a filter. Then we will search this small filter from the top left corner of the matrix. The region where we are searching is called as the receptive field. An important point to note is that the depth of this filter has to be the same as the depth of the input. As the filter is sliding, or **convolving**, around the input image, it is multiplying the values in the filter with the original pixel values of the image. They are added up per reading and stored as a single number. This will represent the value of that stride and will be stored in a new matrix which is called activation map or feature map.

2. **Pooling**

Pooling is a major power tool of CNN to reduce the complexity or size of the array. It is like reducing a 8 megapixel image to a 2 megapixel image without losing any valuable data. For pooling, we have to define pool size. For example, Take pool size as 2, 2. Then for every 2, 2 elements of the matrix, it will be replaced by the highest value in that 2, 2 matrix. After pooling, an image has about a quarter as many pixels as it started with. Because it keeps the maximum value from each window, it preserves the best fits of each feature within the window. This means that it doesn't care so much exactly where the feature fit as long as it fit somewhere within the window. The result of this is that CNNs can find whether a feature is in an image without worrying about where it is.

3. **Activation RELU**

The Rectified Linear Unit or RELU is a small but important player in this process. This helps the CNN stay mathematically healthy by keeping learned values from getting stuck near 0 or blowing up toward infinity. Its math is also very simple, wherever a negative number occurs, swap it with a 0.

4. **Drop Out**

Deep learning neural networks are likely to quickly overfit a training dataset with fewer data. In that case, tweaking the dropout rate will help the model to regain back to a good fit. Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. Some layer outputs are randomly ignored or "dropped out" during training. The dropout hyperparameters default definition is the likelihood of training a given node in a layer, where 1.0 means no dropout, and 0.0 means no layer output. Good value for dropout in a hidden layer is between 0.5 and 0.8.

5. **Fully Connected**

A Fully-Connected layer is a common and cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space. Now that the input image is converted into a suitable form, we shall flatten the image into a column vector. Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model can distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

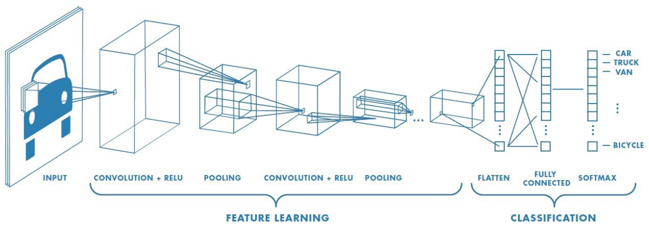

Figure 2. Convolutional Neural Network (CNN) Image Source: medium.com

EXPERIMENTAL EVALUATION AND METHODOLOGY

We tried out various Convolutional Neural Networks whose properties are as follows:

CNN1

Conv1 Filter = 128, Kernel Size= (3, 3)
Pool1 = (2, 2)
Conv2 Filter = 64, Kernel Size= (3, 3)
Pool2 = (2, 2)
Dropout1 = (0.25)
Conv3 Filter = 32, Kernel Size= (3, 3)
Pool3 = (2, 2)
Dropout2 = (0.25)
Flattened()
Dense1 = (128)
Dense2 = (6)

Accuracy: 0.6981481481481482

Averaged F1: 0.6967794691281992

```
           precision   recall  f1-score   support

      0       0.58      0.58     0.58       92
      1       0.70      0.83     0.76       98
      2       0.70      0.68     0.69       91
      3       0.82      0.77     0.79       87
      4       0.71      0.59     0.65       88
      5       0.70      0.74     0.72       84

  accuracy                       0.70      540
  macro avg      0.70     0.70     0.70       540
weighted avg      0.70     0.70     0.70       540
```

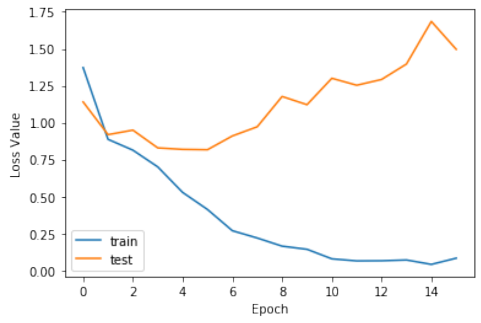

Figure 3. Plot of model Loss on Training and validation dataset for CNN1

CNN2

Conv1 Filter = 32, Kernel Size= (3, 3)
Pool1 = (2, 2)
Conv2 Filter = 64, Kernel Size= (3, 3)
Pool2 = (2, 2)
Dropout1 = (0.25)
Conv3 Filter = 128, Kernel Size= (3, 3)
Pool3 = (2, 2)
Dropout2 = (0.25)
Flattened()
Dense1 = (128)
Dense2 = (6)

Accuracy: 0.7166666666666667

Averaged F1: 0.7022190934763296

```
           precision   recall  f1-score   support

      0       0.80      0.38     0.51       92
      1       0.74      0.76     0.75       98
      2       0.70      0.85     0.77       91
      3       0.66      0.90     0.76       87
      4       0.79      0.55     0.64       88
      5       0.70      0.89     0.79       84

  accuracy                       0.72      540
  macro avg      0.73     0.72     0.70       540
weighted avg      0.73     0.72     0.70       540
```

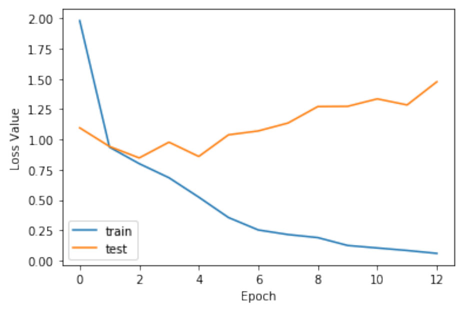

Figure 5. Plot of model Loss on Training and validation dataset for CNN2

CNN3

Conv1 Filter = 32, Kernel Size= (3, 3)
Pool1 = (2, 2)
Dropout1 = (0.25)
Conv2 Filter = 64, Kernel Size= (3, 3)

Pool2 = (2, 2)
Dropout2 = (0.25)
Flattened()
Dense1 = (128)
Dense2 = (6)

Accuracy: 0.7

Averaged F1: 0.6931294828102685

```
          precision   recall  f1-score   support
     0       0.66      0.55      0.60       92
     1       0.68      0.81      0.73       98
     2       0.70      0.76      0.73       91
     3       0.71      0.89      0.79       87
     4       0.73      0.49      0.59       88
     5       0.75      0.70      0.72       84
  accuracy                       0.70      540
  macro avg      0.70      0.70      0.69      540
weighted avg      0.70      0.70      0.69      540
```

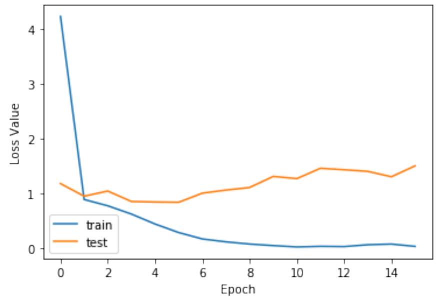

Figure 6. Plot of model Loss on Training and validation dataset for CNN3

CNN4

Conv1 Filter = 32, Kernel Size= (3, 3)
BatchNormalization()
Conv2 Filter = 32, Kernel Size= (3, 3)
BatchNormalization()
Pool1 = (2, 2)
Dropout1 = (0.25)
Conv3 Filter = 64, Kernel Size= (3, 3)
BatchNormalization()
Dropout2 = (0.25)
Conv4 Filter = 128, Kernel Size= (3, 3)
BatchNormalization()
Pool2 = (2, 2)
Dropout3 = (0.25)
Flattened()
Dense1 = (128)
BatchNormalization()
Dropout4 = (0.5)
Dense2 = (64)
BatchNormalization()
Dropout5 = (0.5)
Dense3 = (6)

Accuracy: 0.8222222222222222

Averaged F1: 0.8202781662476992

```
          precision   recall  f1-score   support
     0       0.74      0.85      0.79       92
     1       0.89      0.92      0.90       98
     2       0.82      0.80      0.81       91
     3       0.83      0.97      0.89       87
     4       0.75      0.69      0.72       88
     5       0.94      0.69      0.79       84
  accuracy                       0.82      540
  macro avg      0.83      0.82      0.82      540
weighted avg      0.83      0.82      0.82      540
```

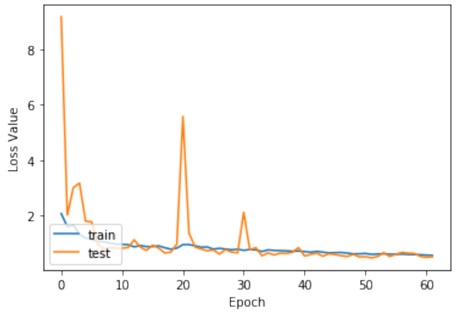

Figure 7. Plot of model Loss on Training and validation dataset for CNN4

For each neuron in the convolutional layer the activation function was RELU.

a. RESULT

We assessed our models performance mainly using the F1 scores and accuracy for the models. We split the data into 80/20 which gave us the optimum result. We used Early Stopping to ensure that our models did not run for too long and to save overfitting our models. We tried different CNN algorithm of which CNN1 had 3 layers with filter values 128, 64, and 32, from the graph we can see that the model is overfitting. We got a 0.69 accuracy on CNN1. Then we tweaked the values of the filter to make the model a good fit. In CNN2, when the value of filters was assigned 32, 64, and 128 the model performed better than CNN1 even though it was overfitting. Then we tried reducing the layers of the CNN which is CNN3. In CNN3, the no of layers was reduced to 2 layers. This model performed well and was closer to a good fit when compared to CNN1 and CNN2, CNN3 gave an accuracy rate of 0.7.

Finally, we tried implementing Batch Normalization with Convoluted Neural Network (CNN4). When data is passed into a deep neural network, the data size can be either reduced of increase to an extreme value. Batch Normalization helps in bringing the value back to the normal range by both mean and variance reference. This had 4 layers of CNN and batch normalization. This model gave an accuracy of 0.82 and the model was in good fit. Hence, we would say CNN4 is the best algorithm.

Below is the confusion matrix for the best model, which is CNN4. So, as we can see that we are getting the dark boxes in the diagonal shape so most of the images are getting classified correctly.

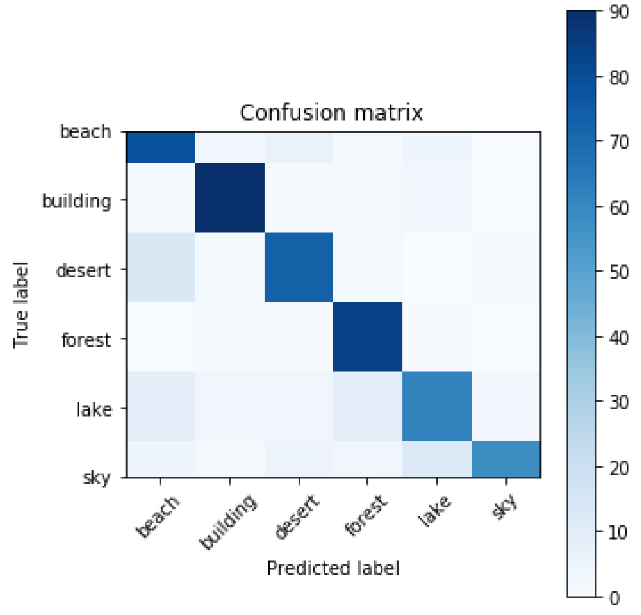

Figure 8. Confusion matrix for the best model (CNN4)

With all the above models we were able to get the result right. We tried inputting an image(Fig below) of a pond inside a forest. The model predicted it right as both forest and lake. The machine also predicted that the closest resemblance as a forest which was the first label and the second label was a lake. The audio of the forest was played thereafter.

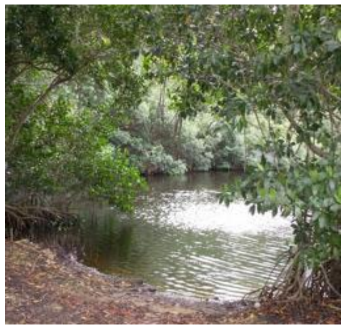

Figure 9. Test image

RELATED WORK

Making machines detect environment was a big challenge to the researchers. [2]. Some researchers from MIT built a dataset just for scene recognition and worked on it, which they recommended that CNN was the optimum algorithm to achieve environment scene detection. The author has used CNN to detect different indoor and outdoor scenes with the accuracy of around 56%.

Petraitis, Taurius & Maskeliunas has mentioned [1] that scenes can be detected using Bag of words approach, the objects in the target image is listed down and then the scene is predicted by using SVM and other algorithms.

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba [4] the author explains about how scene can be detected using SVM. The dataset is same as the dataset we used, which makes our approach similar.

CONCLUSION

Scene recognition is one of the key tasks of computer vision that enables the definition of the significance of object recognition. Four different CNN model were applied to recognize environment from the images. Of all the model CNN4 performed better with an accuracy of 82% and the model was a good fit.

PROJECT APPLICATION

The main idea of the project is to recognize the environment from the given images. This project can be helpful for other projects with a different purpose. That is When a model is created to recognize objects in the pictures, the model has to scan all the objects that could be present in the image. This is when our project comes handy, it can detect the scene of the image, which narrow down categories of the objects, which should be detected from the image. For example, if the detected environment in the first model is buildings, the second model should expect cars, roads, people on the street or in the balcony. Or if the detected environment in the first model is forest then the second model can expect trees, birds, animals, etc.

Secondly, this project can be useful in automation purpose, as in cars to enable some features according to the environments. Or it can be useful in the robots to analyze the environment and act accordingly.

REFERENCES

[1] Petraitis, Taurius & Maskeliunas, Rytis & Damasevicius, Robertas & Połap, Dawid & Woźniak, Marcin & Gabryel, Marcin. (2017). Environment Recognition based on Images using Bag-of-Words. 166-176. 10.5220/0006585601660176.

[2] *Places: A 10 million Image Database for Scene Recognition* B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017*

[3] Rodner, Erik & Hoffman, Judy & Donahue, Jeff & Darrell, Trevor & Saenko, Kate. (2013). Towards Adapting ImageNet to Reality: Scalable Domain Adaptation with Implicit Low-rank Transformations.

[4] "SUN Database: Large-scale Scene Recognition from Abbey to Zoo". J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. IEEE Conference on Computer Vision and Pattern Recognition, 2010.