Pokemon Identifier

Team 30 (Phowa)

Members
Vikas Thamizharasan(201401179)
Adarsh Sanjeev(201401081)
Vinaya Khandelwal(201401139)
Shaleen Garg(201401069)

Mentor
Prateek Singhi

Problem Statement

To make a pokedex(virtual device to help detect pokemon) which on taking an image containing possibly more than one pokemon, as input should be able to locate and recognize the pokemon.(3 classes)

Our Approaches

1. **YOLO** (Convolutional Neural Networks)
    a. Works well for multiple pokemon in one image.
    b. Time consuming to train. Need large amount of annotated images(~200/class).
2. **Multiclass SVM**
    a. Needs very less annotated data(~20/class).

YOLO (Introduction)

Yolo is an approach to object detection using convolutional neural networks. It frames object detection as a regression problem to bounding boxes and respective class probabilities.

- Regression problem allows yolo to have a simple pipeline which makes it extremely fast.
- Yolo takes entire image as input thus using contextual information for prediction which reduces background errors.
- Yolo learns generalizable representations of objects. It Handles unexpected inputs better than R-CNN.

YOLO (Model)

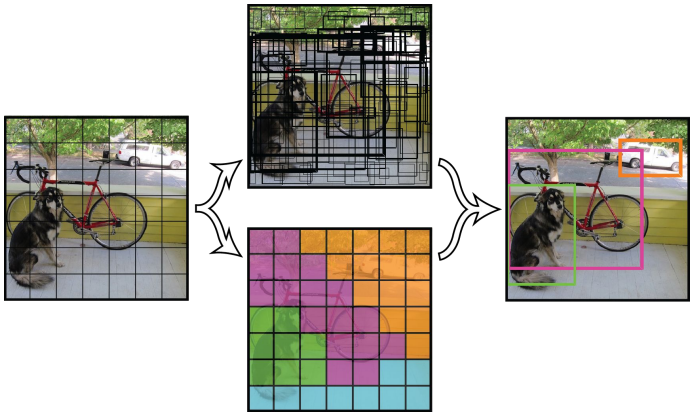
Divide image into SxS grid cells.

Each cell predicts B bounding boxes.

Predicts x, y, width, height & confidence scores for each bounding box

Each Grid cell holds $Pr(Class_i \mid Object)$

Final output = S x S x (B x 5 + C) tensor.



YOLO (Model prediction)

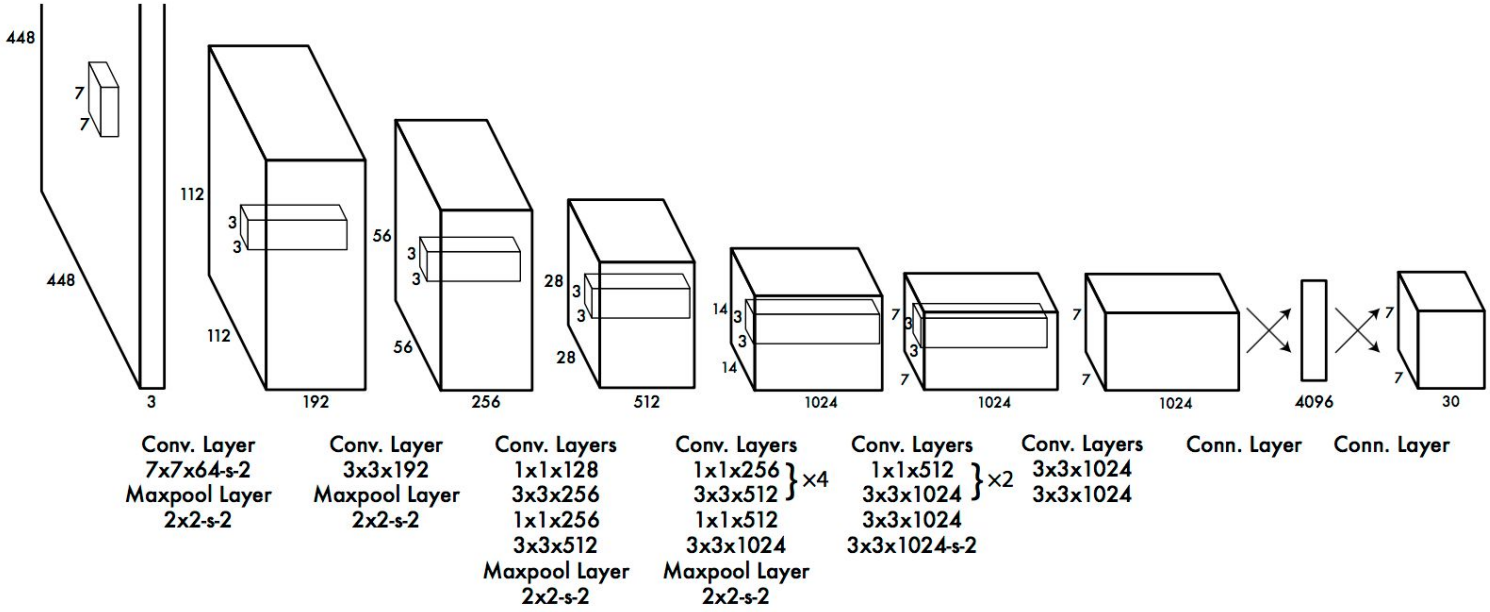
Confidence of Bounding Box = $Pr(Object) \times IOU^{truth/pred}$

Conditional Class Probabilities = $Pr(Class_i \mid Object)$.

Model Prediction =

$$Pr(Class_i \mid Object) \times Pr(Object) \times IOU^{truth/pred} = Pr(Class_i) \times IOU^{truth/pred}$$

YOLO (CNN network architecture)



YOLO (Pokemon Configuration)

Layers used (24 x Convolutional, 4 x Maxpool, 2 x Fully connected)

Convolutional layer used leaky rectified linear activation while fully connected layers used a linear activation function.

Yolo configurations used - Batch size 64, Max_Batch 4000, learning rate 0.0001, Momentum=0.9

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

YOLO (loss function)

λ_{coord} = increase loss from bounding box (5)

λ_{noobj} = decreases loss from classification error (0.5)

$\mathbb{1}_i^{obj}$ = 1 if Object appears in cell i

$\mathbb{1}_{ij}^{obj}$ = 1 if jth bounding box is "responsible" for cell i

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

Problems

1. Overfitting
2. Multiple bounding boxes
3. Unable to detect objects with unusual aspect ratio

Mitigating Overfitting and Multiple Bounding Boxes

1. Decreasing learning rate with increasing epochs
2. Dropout with some probability
3. Data Augmentation (Noise and random Transformations)
4. Adjust HSV

1. Non-Maximal Suppression

Results

Metrics used

$$\text{recall} = \frac{tp}{tp + fn}$$

$$\text{precision} = \frac{tp}{tp + fp} = \frac{tp}{n}$$

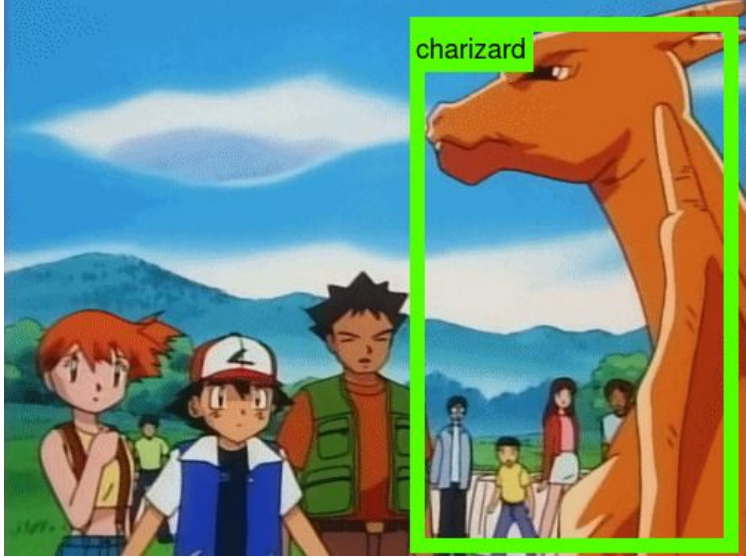
mAP (mean average precision) is used measure of accuracy (58.727%)

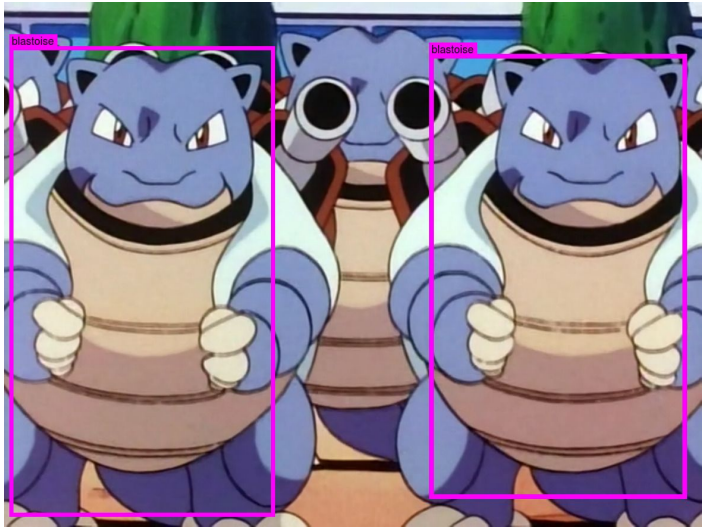
YOLO (Difficulties Faced)

1) Version 2 vs Version 1
2) Scraping data (videos/gifs and images.google.com)
3) Annotating data. Used labelImgGUI to annotate data (xml)

Results


blastoise


charizard


blastoise
blastoise


blastoise
venasaur
VS
BLASTOISE
PC 1336
VENUSAUR
PC 1542

Method #2 : Multiclass SVM

Steps:

1) Convert dataset to grayscale. Primarily to speed up the training process.
2) Resize entire dataset to a fixed dimension(eg 200x200) and then apply PCA to get the eigenvector corresponding to largest eigenvalue. The dataset is resized to fixed dimension so that the input to the classifier are all of the same dimension. From 40000 features (200x200) for each pokemon, it is now 18 after PCA.
3) An SVM classifier is used. **GridSearchCV** function of sklearn is used to run on rbf kernel function with different C and gamma.

```
param_grid = {  'kernel': ['rbf', 'linear'],

'C': [1e3, 5e3, 1e4, 5e4, 1e5],

'gamma': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1], }
```

Method #2 : Using SVM

Results:

Trained on 3 classes of pokemon (Charizard, Pikachu, Tangela) with 20 images per class (Total: 60)

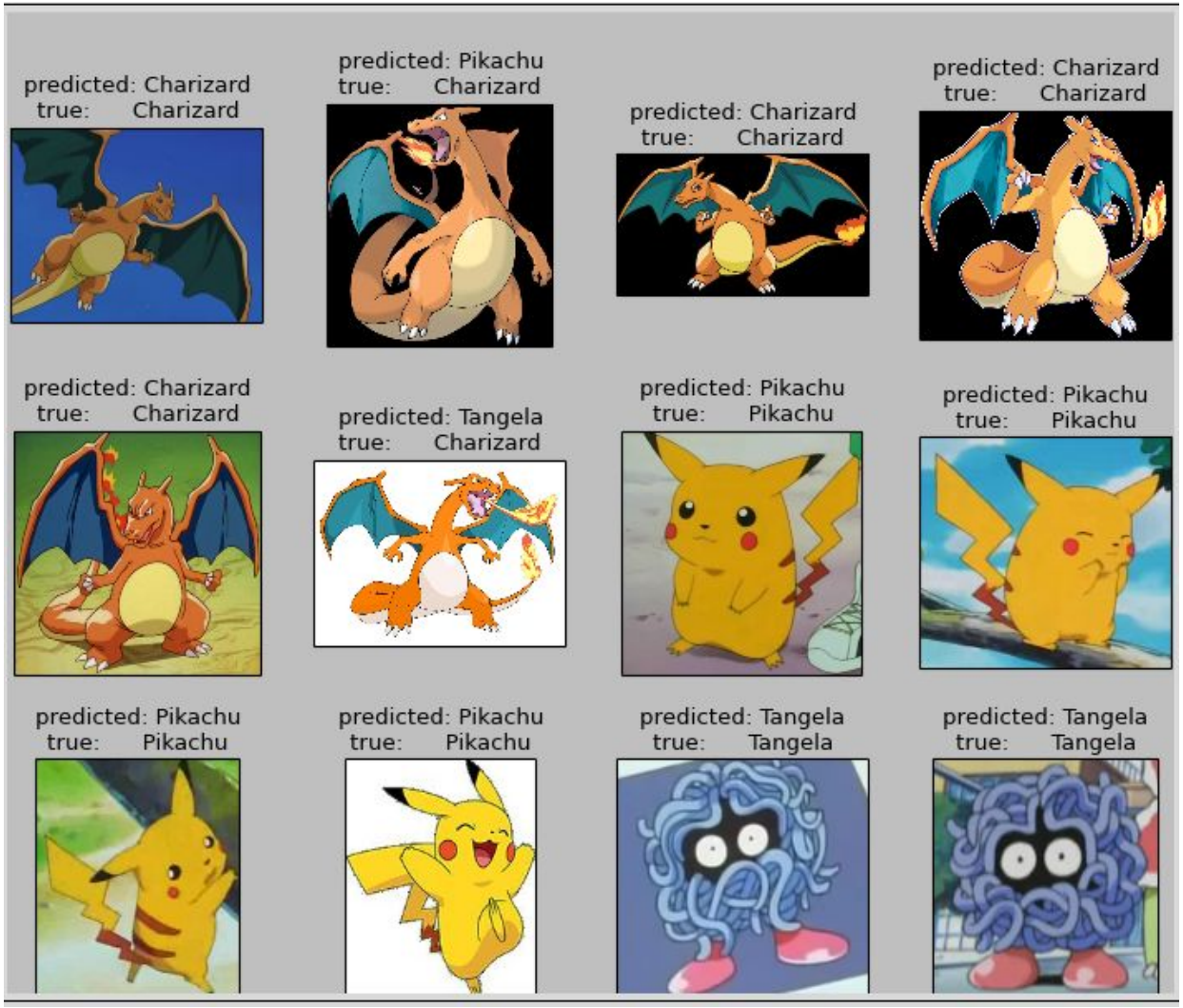
Average accuracy after kfold CrossValidation: **85.17**

Method #2 : Using SVM

Results:

Method #2 : Using SVM

Results:



Method #2 : Using SVM

Advantages:

- Compared to YOLO, this needed a very small dataset and also minimal training time.
- Accuracy is better than YOLO for images with single pokemon.

Disadvantages:

- Cannot handle multiple detections (i.e multiple pokemons in the same image)
- Since the classifier was not trained on negative samples, will classify non pokemon images as one of the classes.

Method #2 : Using SVM

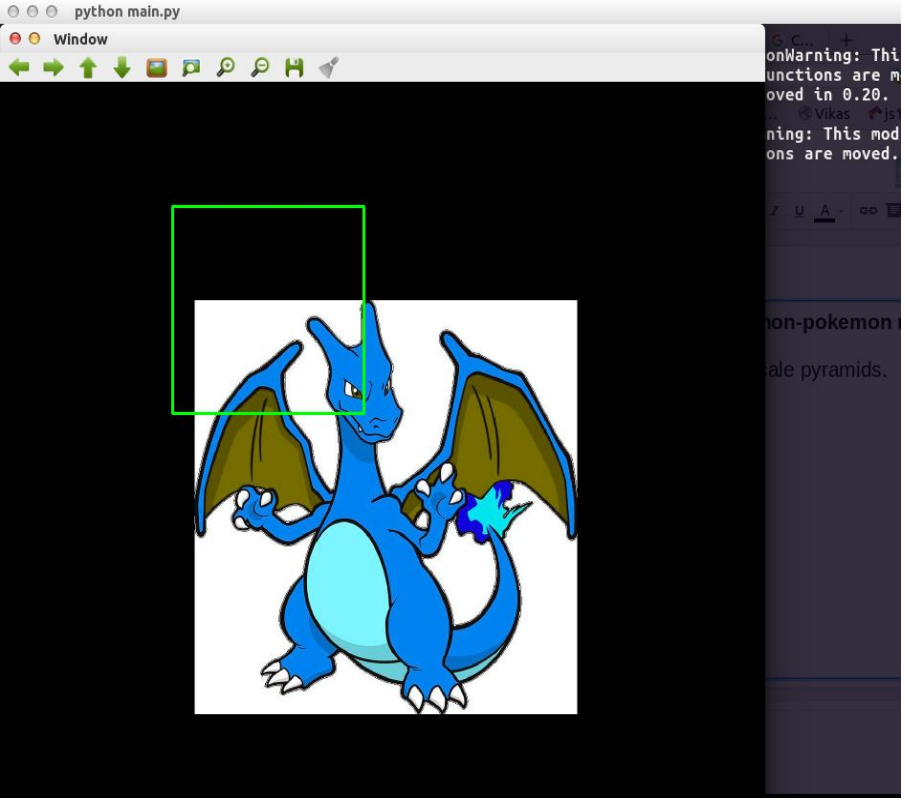
Solution to multiple detection and non-pokemon misclassification

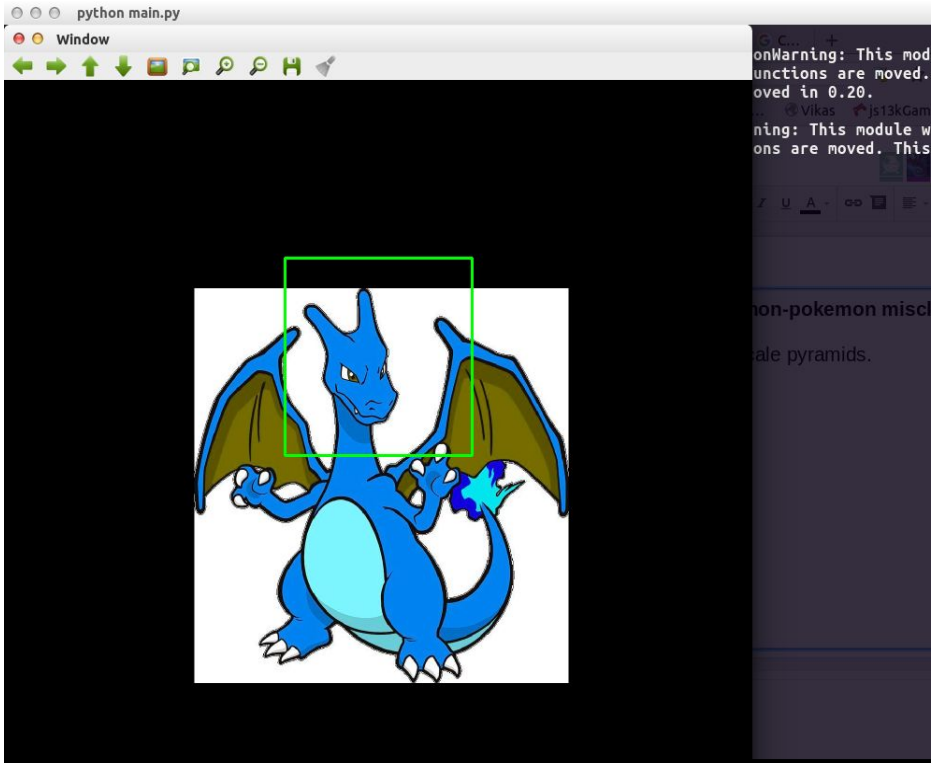
- Using sliding window with multi-scale pyramids.

Idea:

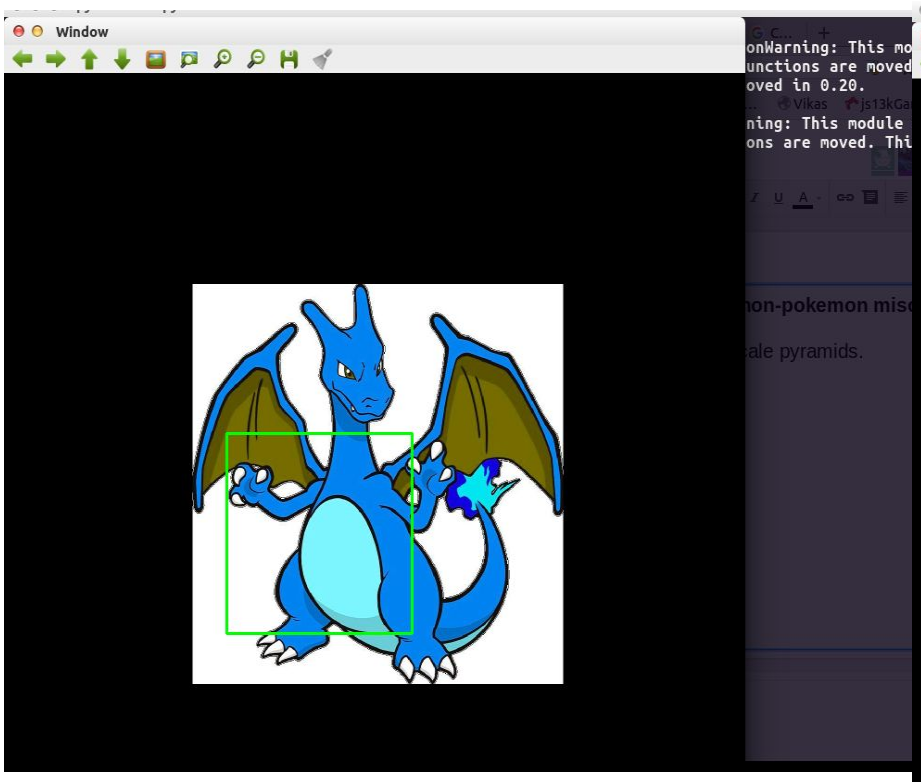
1. Use a fixed sliding window and iterate through the image with a fixed stride.
2. Perform SVM classification for each window. Save bounding box details that had a correct classification.
3. Down scale the image and perform **step 1** and **2** if no correct classifications were found in **step 2** for the previous scale.

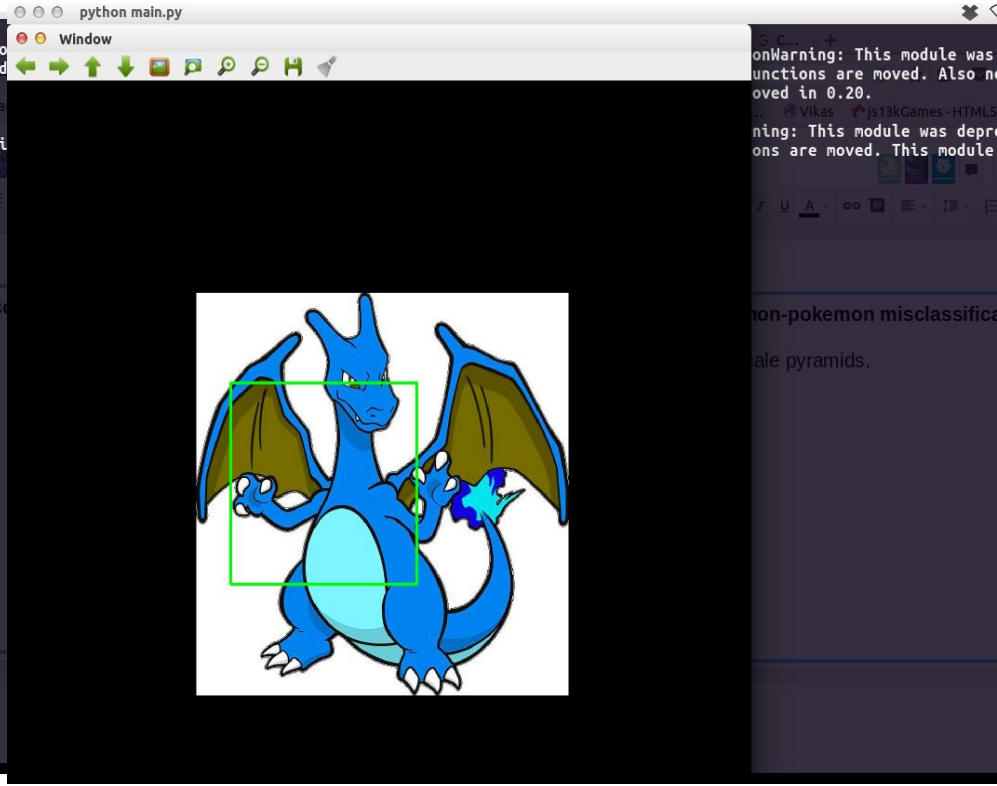
Method #2 : Using sliding window with multi-scale pyramids.



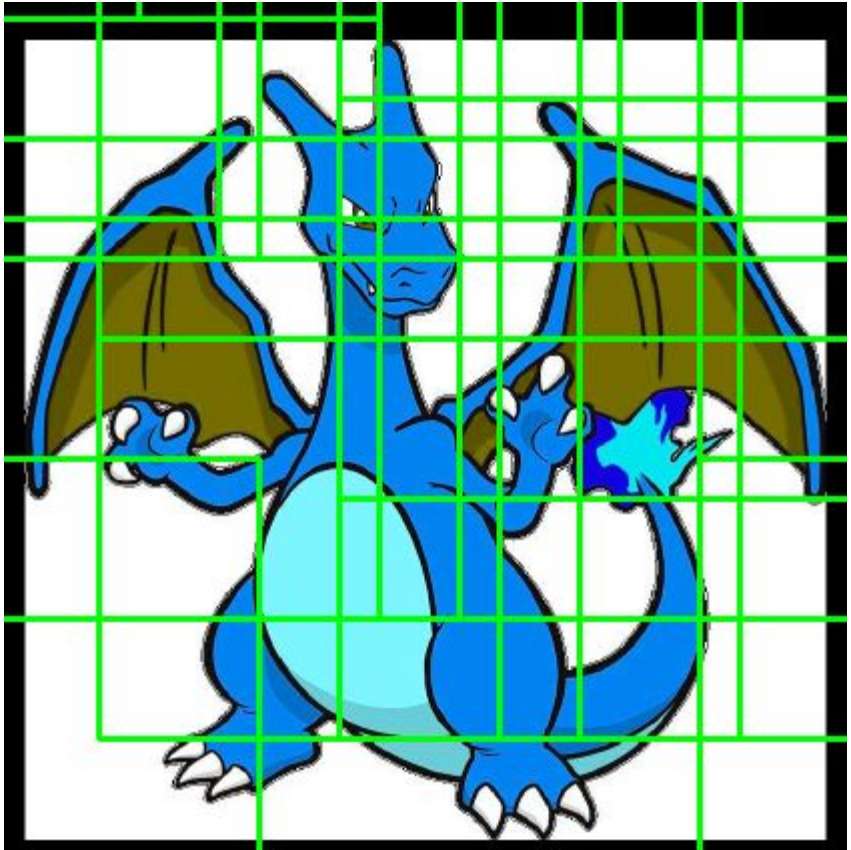


Method #2：Using sliding window with multi-scale pyramids.





Method #2 : Apply NMS (Non-Maximal Suppression)



Method #2 : Using SVM

Solution to non-pokemon misclassification:

- Instead of giving data transformed with PCA to the SVM, use a feature extraction technique like SIFT or feature descriptor like HOG (Histogram of Oriented Gradients) for each image and train the SVM on these.
- This will allow us to create a negative dataset which will be images with few features using SIFT or feature descriptor with mostly zero vector using HOG.
- This will require the data within classes to contain similarity, like mostly face images or frontal poses. This is because the features across image will have to be similar for better classification.

References and Libraries used:

1) https://pjreddie.com/media/files/papers/yolo.pdf
2) https://pjreddie.com/darknet/
3) https://github.com/tzutalin/ImageNet_Utils
4) http://guanghan.info/blog/en/my-works/train-yolo/
5) https://sanchom.wordpress.com/tag/average-precision/
6) https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
7) Nvidia-cuda-toolkit (To make it work on nvidia gpu), python-sklearn

Thank you