Single Object Detection via Bounding Box Regression Using CNN models

Dev Banerjee, Manikanta Sai Manohar Surapathi, Supriya Chigurupati, Agastya Teja Atluri, Sai Charanya Ponnala.

Objective: To predict the coordinates of bounding boxes containing the faces of cats or dogs in images using different CNN architectures.

Desirable Dataset:

Size: The dataset should contain a large number of images that are representative of the real-world objects that the model is expected to detect. The more diverse and comprehensive the dataset is, the better the model is likely to perform.

Annotations: The dataset should have annotations that accurately bound the objects in the images. The annotations should be in a structured data format and must be mappable to the relevant image.

Objects per Image: As per the scope of this project, the dataset should have only one object and a corresponding bounding box annotation per image. Having more than one objects per image would yield inaccurate results with a simple CNN model.

Realistic Variety: The dataset should include realistic variations such as occlusions, clutter, and different lighting conditions to help the model learn to detect objects in real-world scenarios.

Dataset Description:

The dataset selected for this task, the Oxford-IIIT Pets Image dataset is comprised of 7,349 images that depict 37 unique breeds of cats and dogs. The images range in size from 200x200 to 500x500 pixels and are available in both grayscale and color formats. However, only 3685 images were annotated. We implemented a resampling strategy of flipping existing images horizontally and vertically, which allowed us to generate an additional 7370 annotated images.

To calculate coordinates for flipped images, we used a method based on original image coordinates and the flip type (horizontal/vertical). Using diagonal points from annotation files, we defined a rectangle. For horizontal flips, we updated the xmin and xmax values using image width, and for vertical flips, we adjusted the ymin and ymax values using image height. All target bounding boxes were normalized based on image size.

Data Preprocessing:
The images were preprocessed for uniformity by converting them to grayscale and resizing to (224,224). They were then converted to numpy arrays, normalized by dividing by 255, and shuffled into training and validation sets at an 80:20 ratio. The preprocessed images are now suitable for use in training an object detection model.

Models:

Different model architectures have been experimented with to perform the bounding box regression task. However, all architectures have some common features-

1. ReLU activation only- To achieve positive bounding box coordinates, only ReLU activation is used in all layers.
2. All models are built upon a base model described below, with editions of different paddings and dropout layers.

Model architecture- reasoning/thought and expectation

- **Base Model** (BM)

Built as an extremely light version of the VGG16 model. Utilizes some sets of 2 convolution layers and maxpool layers, followed by some dense layers.

```
Model: "sequential_16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_58 (Conv2D)          (None, 222, 222, 32)      320

 conv2d_59 (Conv2D)          (None, 220, 220, 32)      9248

 max_pooling2d_39 (MaxPoolin  (None, 110, 110, 32)     0
 g2D)

 conv2d_60 (Conv2D)          (None, 108, 108, 64)      18496

 conv2d_61 (Conv2D)          (None, 106, 106, 64)      36928

 max_pooling2d_40 (MaxPoolin  (None, 53, 53, 64)       0
 g2D)

 conv2d_62 (Conv2D)          (None, 51, 51, 128)       73856

 conv2d_63 (Conv2D)          (None, 49, 49, 128)       147584

 max_pooling2d_41 (MaxPoolin  (None, 24, 24, 128)      0
 g2D)

 flatten_16 (Flatten)        (None, 73728)             0

 dense_60 (Dense)            (None, 256)               18874624

 dense_61 (Dense)            (None, 128)               32896

 dense_62 (Dense)            (None, 64)                8256

 dense_63 (Dense)            (None, 32)                2080

 dense_64 (Dense)            (None, 4)                 132

=================================================================
Total params: 19,204,420
Trainable params: 19,204,420
Non-trainable params: 0
```

- **Base Model with Padding** (BMP)

The same base structure as above, only with padding added to every convolutional layer. This model can be expected to better perform on edge cases. 19,204,410

- **Base Model with Padding and Convolutional Layer Dropouts** (BMPCD)

BMCD architecture, with padding added in every convolution layer. Can be expected to have improved results when compared to BMCD. 26,020,164

- **Base Model with Padding and Dense Layer Dropouts** (BMPDD)

BMDD architecture, with padding added in every convolution layer. Can be expected to have improved results when compared to BMCD.

- **VGG16 Based Model** (VGGBM)

Convolution base taken from VGG16 model, extended with some dense layers, and final layer modified for bounding box regression. Since VGG16 is a model trained on the huge ImageNET dataset, its convolution layers can be considered as decent feature extractors, which are further used to predict bounding boxes.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 flatten_2 (Flatten)         (None, 25088)             0

 dense_7 (Dense)             (None, 256)               6422784

 dropout_4 (Dropout)         (None, 256)               0

 dense_8 (Dense)             (None, 4)                 1028

=================================================================
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688
```

Comparison metrics

The BM, BMP, BMPCD, BMPDD, and VGBM models are compared on the following basis-

1. Model summary- Number of model parameters

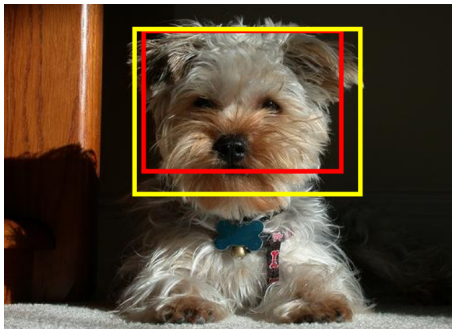
Since all models share the same activation functions, this can be considered a good estimator of model complexity and computation resource requirements.

2. Training curves

Used to assess efficiency of learning and detect overfitting.

3. Results on Validation

Discussion on model predictions on samples from validation dataset. Actual boxes based on target annotations are yellow, predicted boxes are red. For example-



: True Annotation

: Predicted Annotation

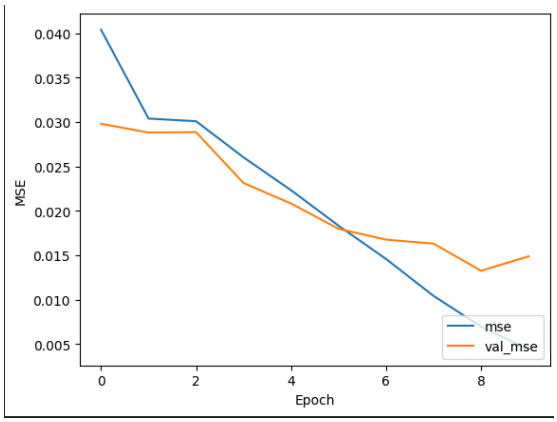
Comparisons

1. **Base Model (BM)** vs **Base Model with Padding (BMP)**

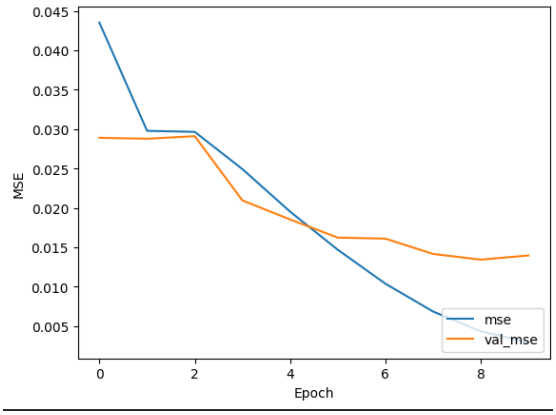
- Number of model parameters

BM - 19,204,420
BMP - 26,020,164

- Training Curves



Base Model (BM)



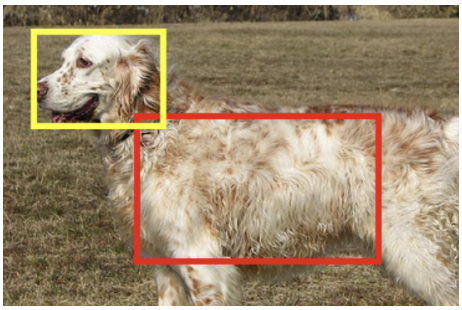
Base Model with Padding (BMP)

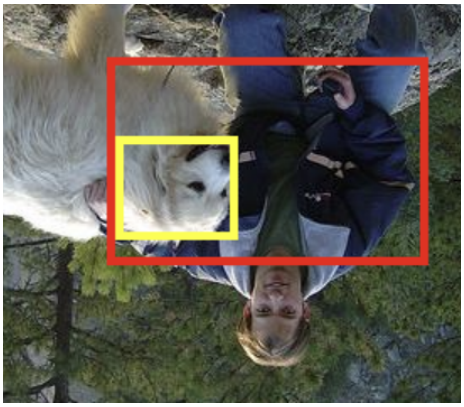
Almost similar epochs to train.

- Validation Results

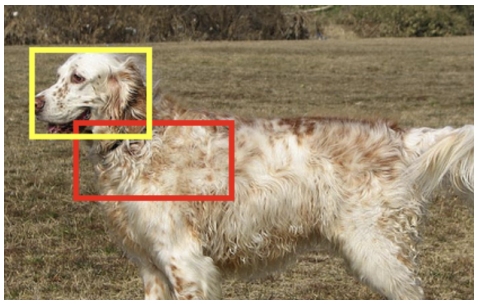
BMP seems to give better localization of objects in general, and hence better results. BMP will be preferred over BM for further comparisons.

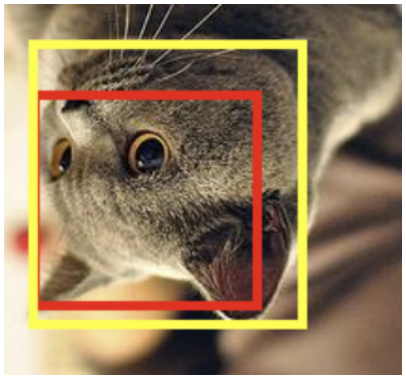
BM





BMP



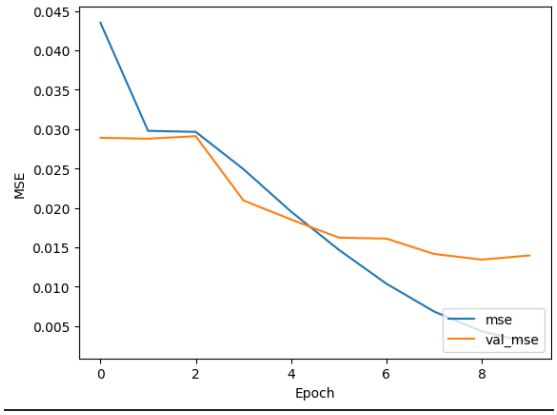


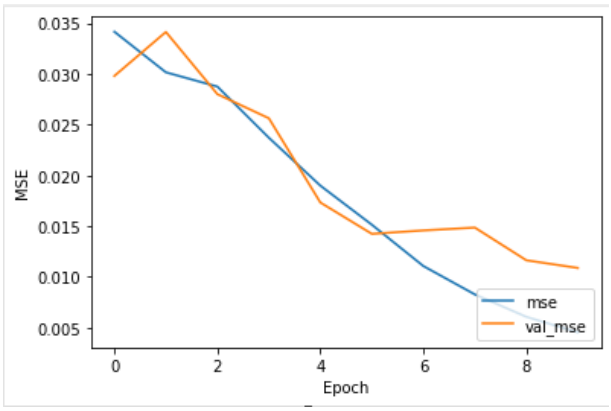
1. **Base Model with Padding (BMP)** vs **Base Model with Padding and Convolutional Layer Dropouts (BMPCD)**

- Number of model parameters

BMP - 26,020,164
BMPCD - 26,020,164

- Training Curves





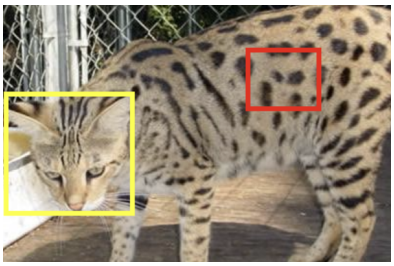
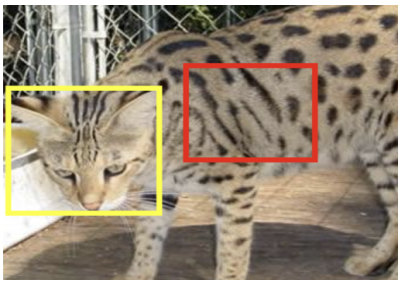
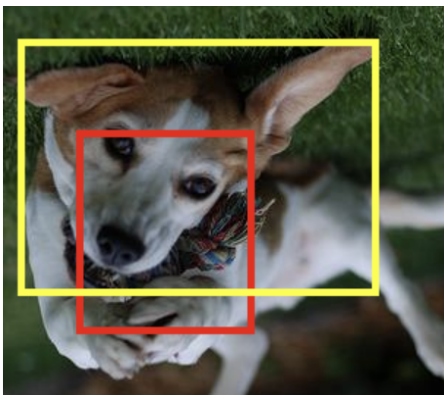
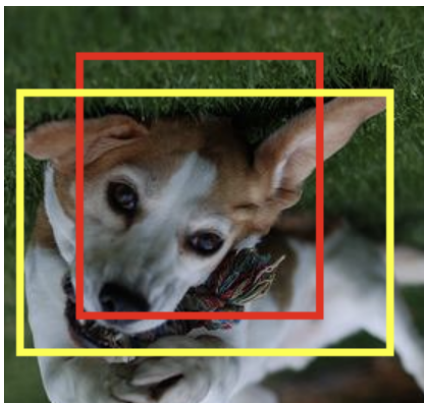
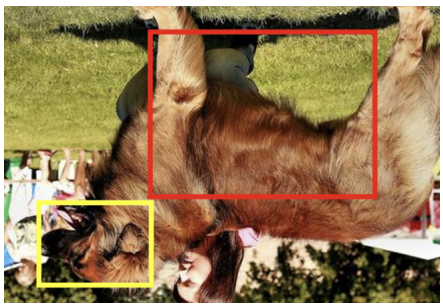
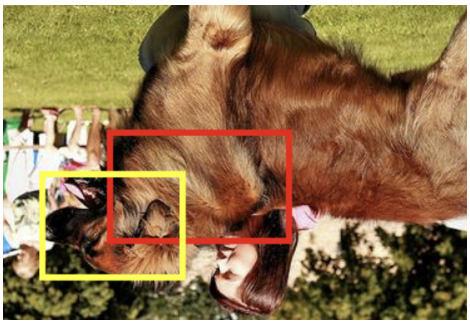
BMP

BMPCD

BMP starts plateauing at 6 epochs. BMPCD learns further beyond 7 epochs. Might be able to learn even beyond 10 epochs, suggesting the model can be built more complicated that its present state.

- Validation Results
  BMPCD is able to predict face boundaries of pets more precisely and accurately than our BMP when presented with challenging images. BMP results tend to accommodate the central part of the image in their predictions, which is regularized in BMPCD results.

BMP **BMPCD**

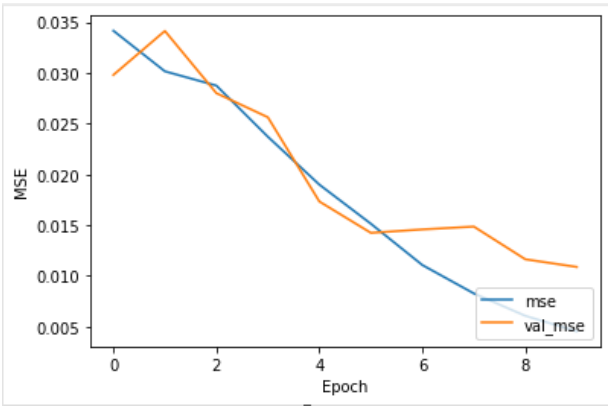







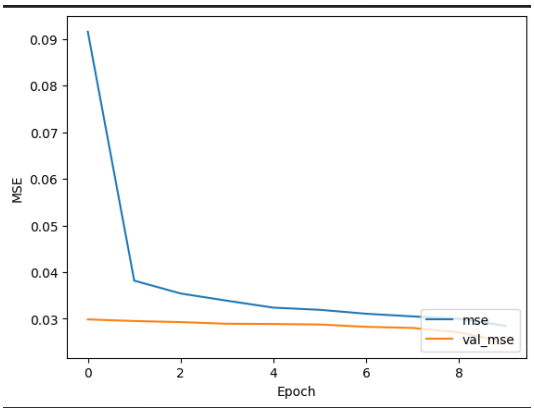
2. **Base Model with Padding and Convolutional Layer Dropouts (BMPCD)** vs **Base Model with Padding and Dense Layer Dropouts (BMPDD)**

- Number of model parameters

BMPCD - 26,020,164
BMPDD - 26,020,164

- Training Curves



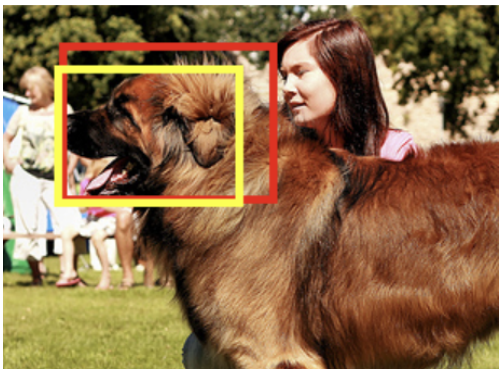
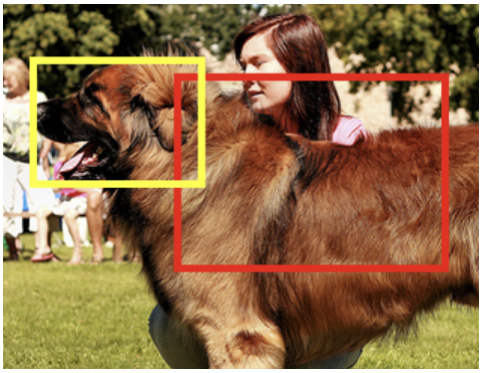
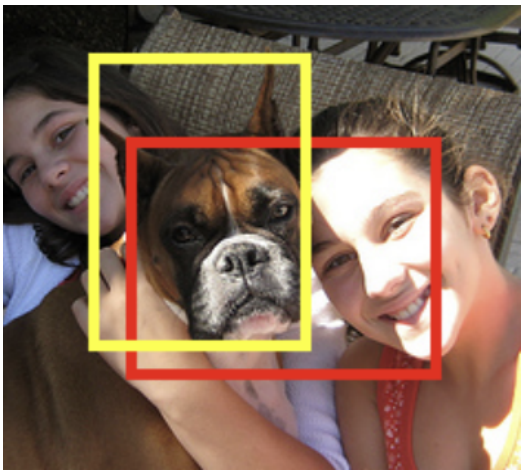
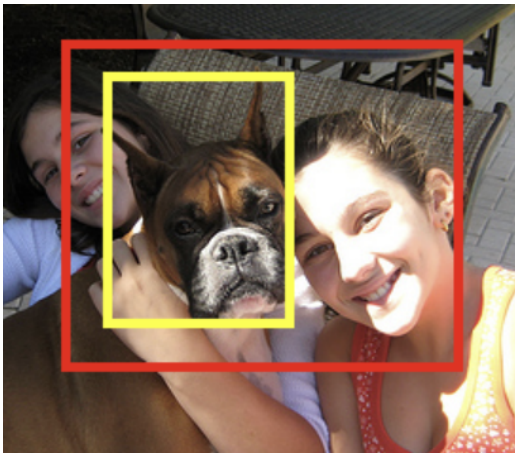
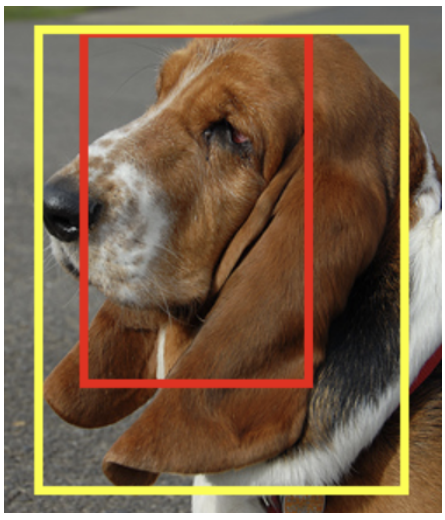


BMPCD **BMPDD**

Training curve for BMPDD does not show much improvement and depicts a typically slow learning model. This can be attributed to the dropout layers being added in the final layers of the network, which introduce noise maybe too late into the network's calculation, thereby generalizing the model too much, and consequently underfitting despite enough training epochs.

- Validation Results: When presented with multiple objects in an image, the BMPCD model is able to predict the boundaries around the pet's face much better than BMPDD. From the below validation images, BMPCD seems to be performing better than BMPDD.

BMPCD **BMPDD**

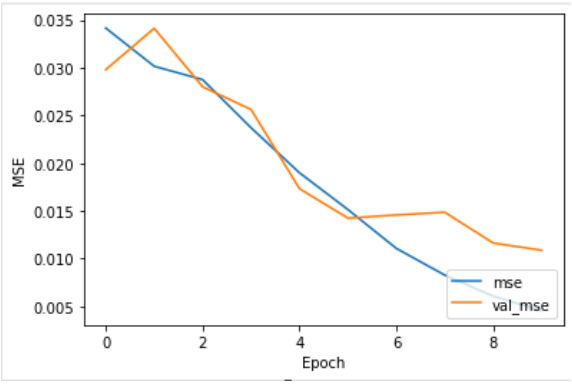

3. **Base Model with Padding and Convolutional Layer Dropouts (BMPCD)** vs **VGG16 Based Model (VGGBM)**

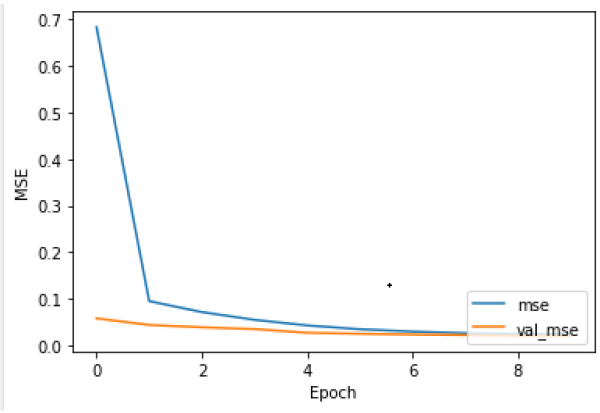
- Number of model parameters

BMPCD - Total params: 26,020,164
Trainable params: 26,020,164
Non-trainable params: 0

VGGBM -
Total params: 21,138,500
Trainable params: 6,423,812
Non-trainable params: 14,714,688

- Training Curves

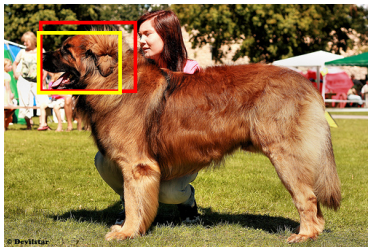


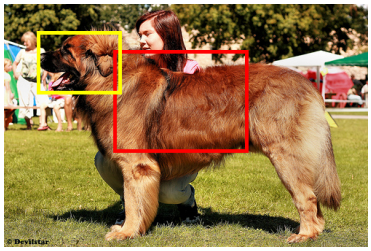


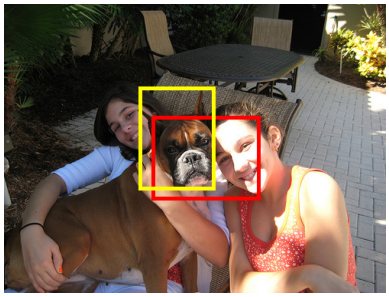
BMPCD **VGGBM**

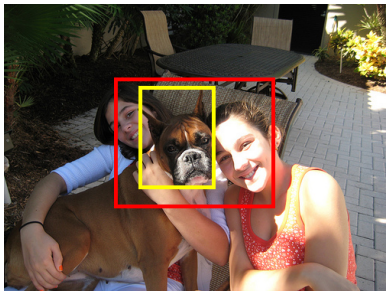
Since there are not many trainable parameters in the VGG16 based model, the model converges to an almost constant validation error quickly. However, as the epochs proceed, the validation error plateaus to a value greater than BMPCD's error.

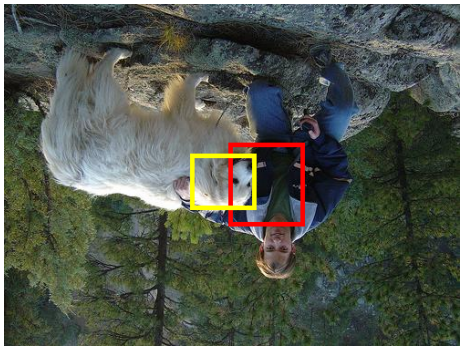
- Validation Results

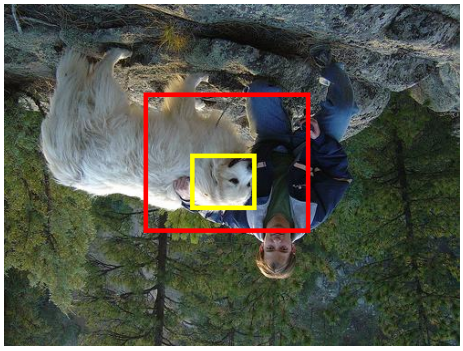












Conclusion

Base Model with Padding and Convolutional Layer Dropouts (BMPCD) has the best results on validation data for the given dataset. The model correctly predicts the object bounding boxes for most cases of single cat or dog images.

Closing notes

- It was seen that images resampled by flipping about vertical or horizontal axis are regarded as an entirely new image by the model, and is hence a decent way to resample the data if needed.
- Regularization in dense layers alone did not produce good results for bounding box regression.
- In the case of single object bounding box regression, it is preferable to have convolution layers with padding, and dropout layers in the (convolution/dense) region of the network.
- Padding layers seem to add flexibility in the location of prediction, and dropouts seem to improve localization of objects
- The modified VGG16 model perform seems to give modest results, however fine tuning the architecture beyond the convolution base may give promising results.
- Better models, built specifically for this purpose, for eg YOLO, can be explored, which utilize techniques other than just CNN.