Multi-Label Image Classification

Patrick Myers, Gaurav Jindal
Sanchit Sinha, Rishab Bamrara

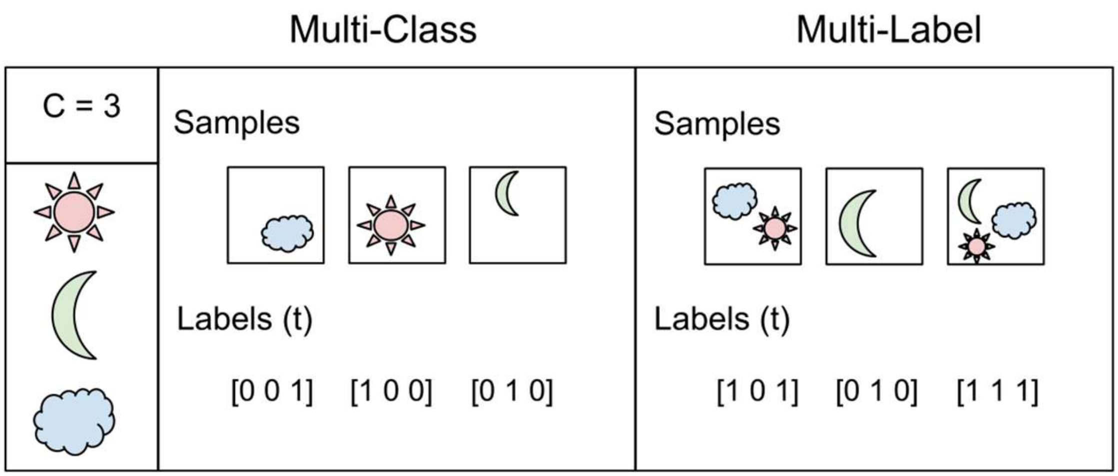
1

Introduction:

- Supervised learning task involving prediction of one or more correct labels for an image.
- Basically a generalisation of multi class classification where multiple labels may be assigned to each images.
- Most of the real life pictures have multiple labels so it is very important to be able to perform multi label classification.
- Images with multiple objects need to be labelled according to the different objects present in it.

2

Problem Statement:



https://gombru.github.io/2018/05/23/cross_entropy_loss/

3

Past Work:

Multi-label Image classification is a well studied problem. Previous SOTAs have used standalone CNNs, fusion of CNNs+RNNs and Graph CNNs.

1. CNN-RNN: A Unified Framework for Multi-label Image Classification, Wang et al.
2. Multi-label image recognition by recurrently discovering attentional regions, Wang et al.
3. Multi-Label Image Recognition with Graph Convolutional Networks, Chen et al.

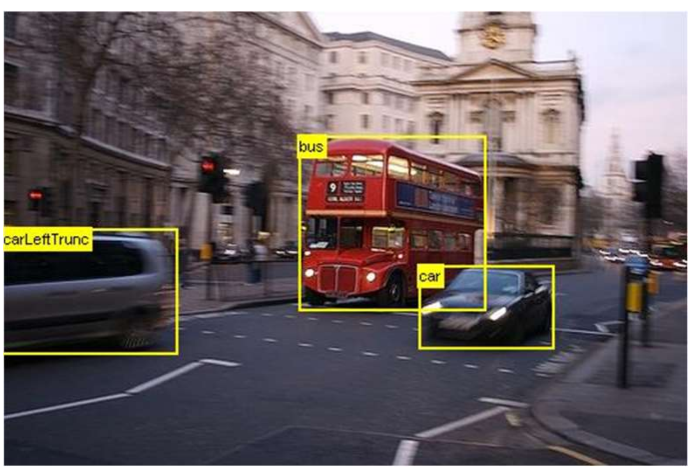
4

PASCAL Visual Object Challenge Dataset (2007):

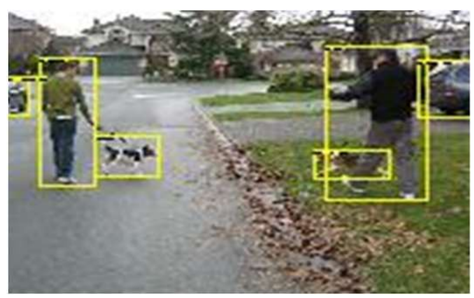
- 9963 images with 20 class labels
- Classes include Aeroplanes, Bicycles, Birds, Cars, Horses, People, TV, etc.
- Built for multilabel image classification

5

Data Example:



Label: Car and Bus



Label: Cat and People

Label: Horse and People

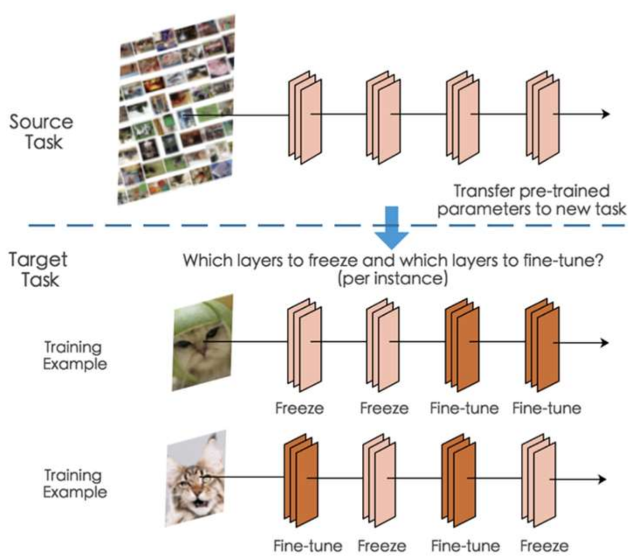
6

Approach:

- Experimented with several existing state-of-the-art models:
    - Alexnet
    - ResNet
    - Googlenet
- Used transfer learning (fine-tuning)
- Tuned hyperparameters (learning rate, batch size)
- Ran each model for 10 epochs and compared results

7

Transfer Learning



https://www.ibm.com/blogs/research/2019/06/spottune-transfer-learning/

8

Data Preprocessing:

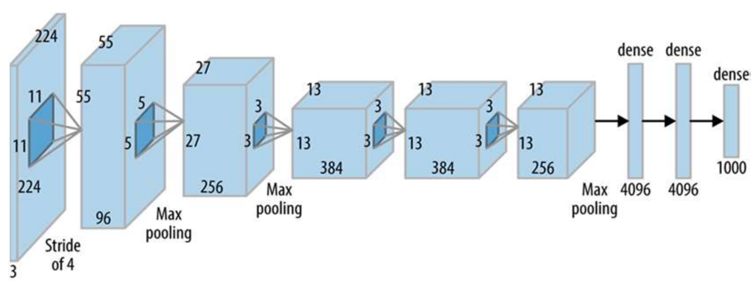
- Image Transformations:
  - Resize the images into 256 x 256 pixels
  - Normalized the images with mean and standard deviation
- Label Transformations:
  - Multi-Hot Encoding

9

Model Architecture:

We have tried the following 3 models:

1. AlexNet:
2. Googlenet:
3. ResNet 18:



Hyperparameters:

- Train-test split: 70%
- Batch size: 20
- Number of epochs: 10
- Probability threshold: 0.5
- Optimiser: SGD
- Learning rate: 0.01
- Momentum: 0.9

https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

10

Loss Function Choice:

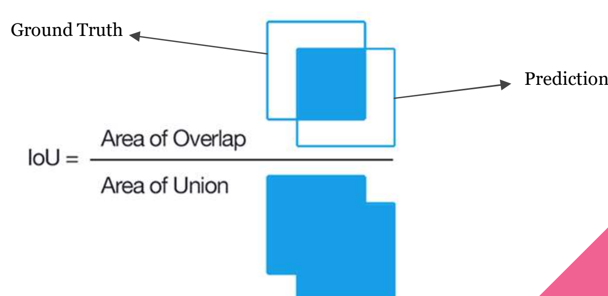
- Our loss function is **Binary cross entropy with logits**
- BCEWithLogitsLoss:

$$BCE = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot log(\hat{y}_i) + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

11

Accuracy Metrics:

1. **Precision:** Ratio of correctly predicted positive observations to the total predicted positive observations.
2. **Recall:** Ratio of correctly predicted positive observations to the all observations in actual class.
3. **F1 Score:** Weighted average of Precision and Recall.
4. **Intersection over Union (IoU):** Often used in object detection challenges.

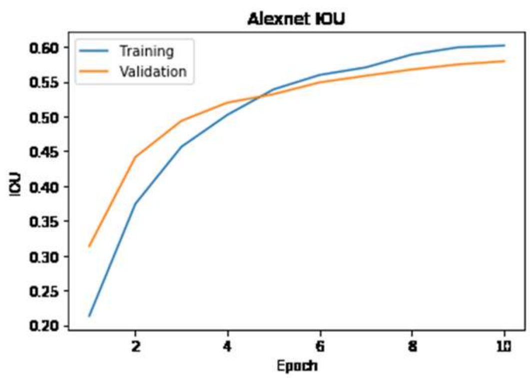


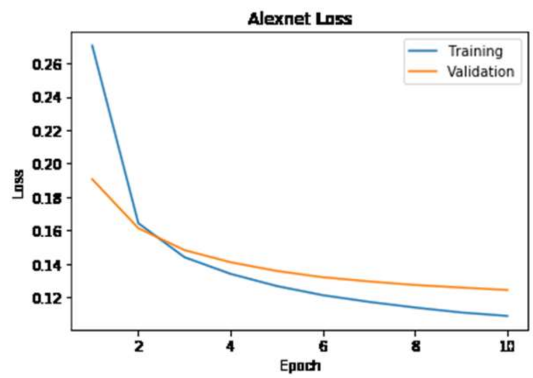
https://www.pyimagesearch.com

12

Results:

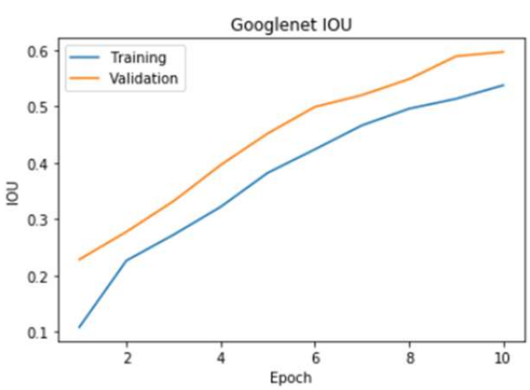
Alexnet

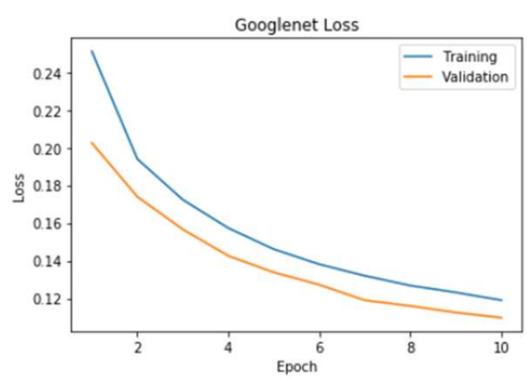




13

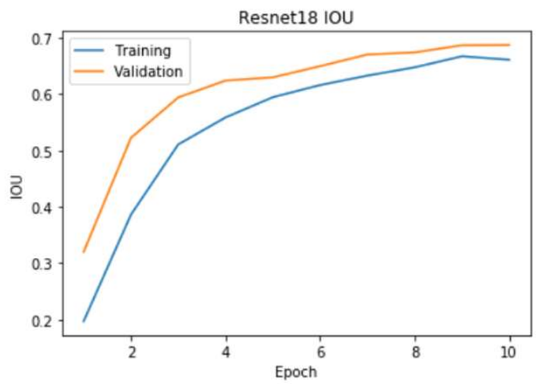
Googlenet

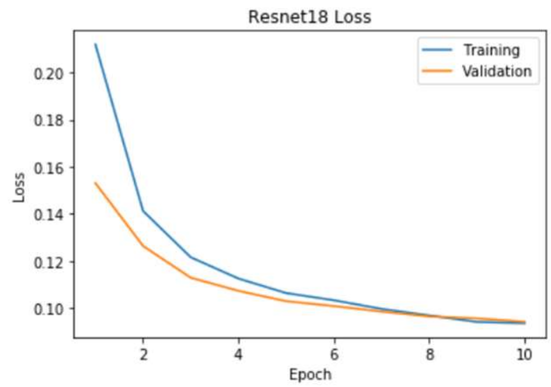




14

Resnet18





15

Validation Results:

Model	IOU	Precision	Recall	F1 Score
AlexNet	0.5793523809523812	0.6982666666666666	0.6255222222222223	0.6255222222222223
Resnet18	0.6871396396396401	0.8081644144144142	0.7220720720720721	0.7220720720720721
GoogleNet	0.5962274774774782	0.7188063063063062	0.6130518018018024	0.6130518018018024

16

References:

1. https://pyimagesearch.com
2. VOC 2007 Dataset: http://host.robots.ox.ac.uk/pascal/VOC/voc2007/
3. PyTorch: http://pytorch.org/
4. CNN-RNN: https://arxiv.org/abs/1604.04573
5. Non DL: https://arxiv.org/pdf/1702.01460v5
6. Graph CNN: https://arxiv.org/pdf/1904.03582v1.pdf

17

18

In [12]:
```
# Hyperparameters
epochs = 10
threshold = 0.5
learning_rate = 0.01
momentum = 0.9
selected_model = "Googlenet"
```

In [13]:
```
if selected_model == "Alexnet":
    # Alexnet
    model = models.alexnet(pretrained=True)

    for parameter in model.parameters():
        parameter.requires_grad = False

    fc_features = model.classifier[-1].in_features
    model.classifier[-1] = nn.Linear(fc_features, num_classes)
    classifier = model.to(torch.device("cuda:0"))

    loss_function = nn.BCEWithLogitsLoss()
    optimizer = torch.optim.SGD(classifier.classifier[-1].parameters(), lr=learning_rate, momentum=momentu
m)

elif selected_model == "Resnet18":
    # Resnet18
    model = models.resnet18(pretrained=True)

    for parameter in model.parameters():
        parameter.requires_grad = False

    fc_features = model.fc.in_features
    model.fc = nn.Linear(fc_features, num_classes)
    classifier = model.to(torch.device("cuda:0"))

    loss_function = nn.BCEWithLogitsLoss()
    optimizer = torch.optim.SGD(classifier.fc.parameters(), lr=learning_rate, momentum=momentum)

elif selected_model == "Googlenet":
    # Inception V3
    model = models.googlenet(pretrained=True)

    for parameter in model.parameters():
        parameter.requires_grad = False

    fc_features = model.fc.in_features
    model.fc = nn.Linear(fc_features, num_classes)
    classifier = model.to(torch.device("cuda:0"))

    loss_function = nn.BCEWithLogitsLoss()
    optimizer = torch.optim.SGD(classifier.fc.parameters(), lr=learning_rate, momentum=momentum)
```

19

In [14]:
```
training_losses = []
training_ious = []
validation_losses = []
validation_ious = []
print("Training on", selected_model)
print("Hyperparameters:\n",
      "\ntrain_split:", train_split,
      "\nbatch_size:", batch_size,
      "\nepochs:", epochs,
      "\nthreshold:", threshold,
      "\nlearning_rate:", learning_rate,
      "\nmomentum:", momentum,
      "\nselected_model:", selected_model,
      "\nloss_function:", "BCEWithLogitsLoss",
     )
for epoch in range(0, epochs):
    print("\nstarting epoch:", epoch + 1)
    training_loss, training_precision, training_recall, training_f1_score, training_iou = train(train_data
_loader, classifier, loss_function, optimizer, threshold)
    training_losses.append(training_loss)
    training_ious.append(training_iou)
    print(selected_model, "results for epoch", epoch + 1)
    print("training_loss:", training_loss)
    print("training_iou:", training_iou)
    print("training_recall:", training_recall)
    print("training_precision:", training_precision)
    print("training_f1_score:", training_recall, "\n")

    validation_loss, validation_precision, validation_recall, validation_f1_score, validation_iou  = valid
ate(validation_data_loader, classifier, loss_function, threshold)
    validation_losses.append(validation_loss)
    validation_ious.append(validation_iou)
    print("validation_loss:", validation_loss)
    print("validation_iou:", validation_iou)
    print("validation_recall:", validation_recall)
    print("validation_precision:", validation_precision)
    print("validation_f1_score:", validation_recall, "\n\n")
```

```
Training on Googlenet
Hyperparameters:

train_split: 0.7
batch_size: 20
epochs: 10
threshold: 0.5
learning_rate: 0.01
momentum: 0.9
selected_model: Googlenet
loss_function: BCEWithLogitsLoss

starting epoch: 1
Googlenet results for epoch 1
training_loss: 0.2510858216898979
training_iou: 0.1085934044279418 8
training_recall: 0.12061175337186877
training_precision: 0.1727353731688992
training_f1_score: 0.12061175337186877

validation_loss: 0.2025171745870564 5
validation_iou: 0.2285135135135134
validation_recall: 0.2285135135135134
validation_precision: 0.3574324324324324 5
validation_f1_score: 0.2285135135135134
```

20