GINA CODY
SCHOOL OF ENGINEERING
AND COMPUTER SCIENCE

CLASSIFICATION OF CHEST X-RAY IMAGES TO DISTINGUISH BETWEEN COVID-19 AND PNEUMONIA

COMP 6721 - APPLIED ARTIFICIAL INTELLIGENCE

GROUP - D

Purpose

- Pandemic causing severe acute respiratory syndrome.
- Identify Covid from numerous similar cases of Pneumonia, Tuberculosis, Lung infections etc.
- Correct and early diagnosis.

Proposal

- Proposing CNN models to differentiate classes.
- Training the data.
- Study the impact of different training models in our application by interchanging the datasets, CNN architectures, hyper parameters etc.

Data?

	#images	#classes	Image_format	Image_size	Source	Average Size
Dataset 1	21,164	4	PNG	299*299	Kaggle	1,345
Dataset 2	6,939	3	PNG	1024*1024	Kaggle	2,133
Dataset 3	654	27	PNG	Random	Github	-

Dataset 1

"COVID-19 Radiography Database"

- COVID - 3616 Images
- Normal - 10,192 Images
- Non-COVID lung infection - 6012 Images
- Pneumonia - 1345 Images

Dataset 2

"COVID19, Pneumonia, Normal Chest Xray Dataset"

- COVID – 2,133 Images
- Normal – 2,133 Images
- Pneumonia - 2,133 Images

Dataset 3

"Covid-chestxray-dataset"

"

- Train class - 481 images
- Test class – 149 images
- Multi-label images

Change in Dataset 3

- Earlier we were training a dataset which has 22 classes, but the image data was not labeled so we needed to change the dataset then we shifted our dataset 3 to new dataset which has 27 classes, and we were able to train our model perfectly on that dataset.
- New Dataset had classes as: Covid-19, Chlamydophila, Fungal, Influenza, etc.

Class : Pneumonia

Class : Tuberculosis

Class : Covid-19

Class : Tuberculosis

CNN Models:

- AlexNet
- VGG11
- ResNet50

Pre-Processing

Enhance image features

Images are of different sizes and formats

Prepare uniform data for training :

- Center Crop, Horizontal Flip
- Normalize images with mean
- Resize image size : 224*224

Training Parameters:

To train our models we have used:

- Batch size : 32
- Loss function : Cross Entropy
- Learning rate : 0.0001
- Epochs : 50

After Training:

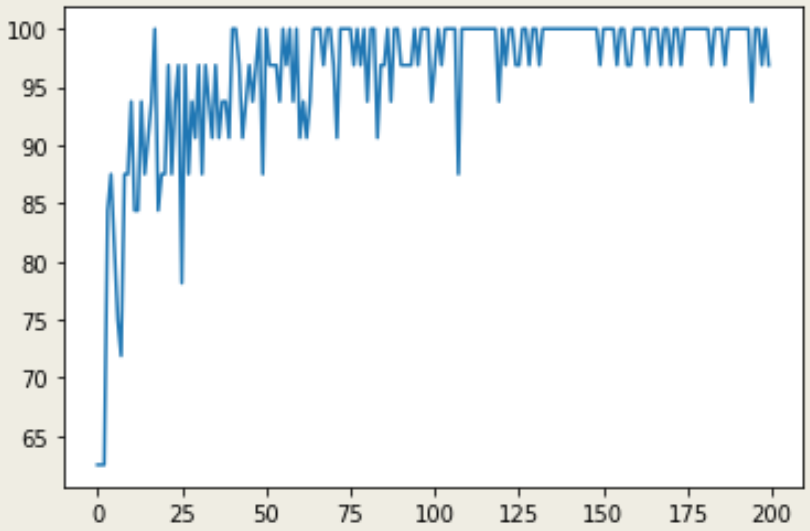
- After training models for 50 epochs, we are evaluating model and getting the accuracies.
- To check the performance, we are also getting Precision Recall, F1 score, and Confusion matrix.

Results?

	Accuracy			Precision			Recall			F1 score		
Datasets/ Models	1	2	3	1	2	3	1	2	3	1	2	3
AlexNet	91.51	93.12	77.54	0.9361	0.9351	0.2143	0.9106	0.9317	0.1148	0.9152	0.9313	0.2692
VGG11	94.36	95.48	75.03	0.9461	0.9568	0.0815	0.9550	0.9549	0.0492	0.8880	0.9548	0.1154
ResNet50	90.94	94.26	64.02	0.9090	0.9451	0.1945	0.9236	0.9429	0.1115	0.9095	0.9426	0.2615

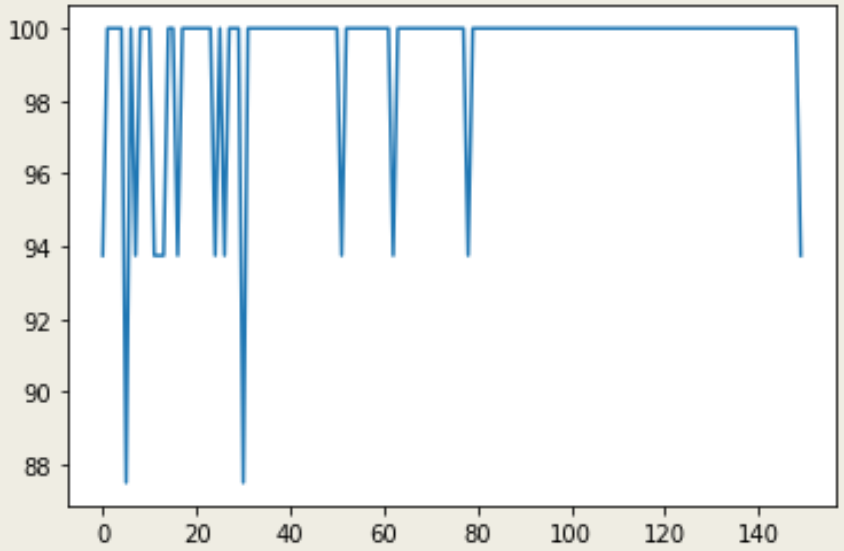
Top Accuracies: VGG Model

- Chosen Model: VGG11



Accuracy : 91.93%

Dataset : 1

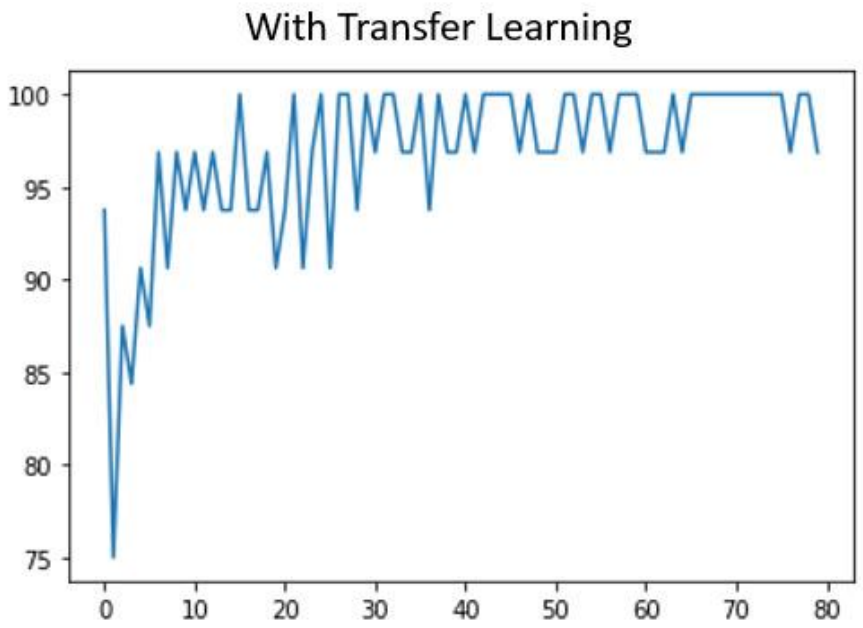


Accuracy : 93.93%

Dataset : 2

Transfer Learning

- Chosen Model: VGG11



Total Accuracy : 94.36%

Total Accuracy : 88.36%

Results of Dataset 1

Hyperparameter Tuning

- Batch Size

```
__all__ = [
    "VGG",
    "vgg11", "vgg13", "vgg16", "vgg19",
    "vgg11_bn", "vgg13_bn", "vgg16_bn", "vgg19_bn",
]

vgg_cfgs: Dict[str, List[Union[str, int]]] = {
    "vgg11": [64, "M", 128, "M", 256, 256, "M", 512, 512, "M", 512, 512, "M"],
    "vgg13": [64, 64, "M", 128, 128, "M", 256, 256, "M", 512, 512, "M", 512, 512, "M"],
    "vgg16": [64, 64, "M", 128, 128, "M", 256, 256, 256, "M", 512, 512, 512, "M", 512, 512, 512, "M"],
    "vgg19": [64, 64, "M", 128, 128, "M", 256, 256, 256, 256, "M", 512, 512, 512, 512, "M", 512, 512, 512, 512, "M"],
}
```

- Learning Rate

```
# model = vgg11();
model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg11', pretrained=True)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg11_bn', pretrained=Tr
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg13', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg13_bn', pretrained=Tr
```

```
3]:
# model = vgg11();
model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg11', pretrained=False)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg11_bn', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg13', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'vgg13_bn', pretrained=True)
```