Task 1:Detection of Pneumonia in Medical Images.

Sadanand Vithal Giraddi
School of Computer Science
University of Lincoln
Lincoln, United Kingdom
25895786@students.lincoln.ac.uk

***Abstract*— In this assignment, the machine learning techniques such as convolutional neural networks (CNN) were performed on the detection of pneumonia in medical images, and various transfer learning techniques of classifiers like VGG16, ResNet50, and DenseNet121were performed on the detection of pneumonia in medical images and predicted the pneumonia is there or not by giving the input image. And performance is investigated based on the classifiers which include matrices such as TP/FP rates, F-measure, Precision-Recall, and accuracy of the model is also validated by comparing with the other classifiers and noted in the report. The Implementation is done using the Keras module of the TensorFlow framework in the python script.**

Keywords*— *Convolution Neural Network (CNN), Transfer learning techniques, pneumonia detection, VGG16, DenseNet121, ResNet50.

I. Introduction

Pneumonia: Pneumonia is a respiratory infection that affects the lungs which are caused by bacteria, viruses, and fungi [1]. Over the world, year-by-year people are diagnosed with pneumonia and it is also seen that it affects children and individuals over the age of 60 [1]. The machine learning techniques are used for detecting pneumonia from the MRI or CT scan images in the computer for developing the advancement of machine learning by making the computers read the data and preprocess the data by using different techniques like VGG16, ResNet50, CNN, DenseNet121, etc. are used for the feature extraction in the health care field. The datasets are provided by the Kaggle competition of the Pneumonia detection challenge.

Convolution Neural Networks and Transfer learning Techniques: CNN consists set of learning filters which are called kernels and the weights are shared in the particular layer which makes them more efficient in using forward and backward propagation [2]. The layers are arranged in 3 dimensions they are height, width, and depth. Each layer transforms 3D input to the 3Doutput volume for activation of neurons. Padding tells how the border in the sample is handled. And we are using the Activation function as a Rectified linear unit (Relu) [2]. The pooling layer act as an extra layer which is added after the convolution layer [2]. Max pooling is used to calculate the maximum value of the patch and average pooling are used to calculate the average value for the patch. And Fully convolution layers are simple and all the weights in the CNN can be trained by the Backpropagation methods. The softmax function is the normalized function of exponential [2]. CNN has gained deeper attention for the classification of disease [3].
The different transfer learning techniques used in this are DenseNet121, VGG16, and ResNet50 for training the models on the dataset provided in the Assessment are customized to less number of images in which the training images consist of 3000 images with two different classifications i.e., pneumonia and no-pneumonia and test and Val path contains 500 images each.
The original dataset can be found in the following Kaggle competition: https://www.kaggle.com/c/rsna-pneumonia-detection-challenge.
The transfer learning classifiers like VGG16, DenseNet121 and ResNet50 are used for building faster and more effective models of classifying the images [4].

II. Implementations of Steps

1) Importing the CNN, VGG16, ResNet50, and DenseNet 121 Libraries using TensorFlow Keras libraries in the jupyter notebook. The dataset is loaded by giving the exact path of the data and resizing the image to [224,224].
2) Preprocessing the data using CNN, and Transfer learning techniques like VGG16, ResNet50 and DenseNet50 models and also processing the layers of each technique. In DenseNet121 model the model is trained by adding the Flatten layer and Dense layer of 1000 and added the Dropout of 0.25. For CNN model: It Consists of four convolution layers with flatten layer and added the dropout (0.25) overall for the convolution layers and consist of four dense layers (256,128,64,2). CNN model [Altered layers]: It consist of three convolution layer with one drop out(0.25) and a dense layer of (128, 64,10). ResNet50 Model: The extra flatten layer is used with the dense layer (1000). VGG16 model: These are the pre-trained models. We added four convolution layers with flatten layer and the dense layer (1000). ResNet50 Model [altered layers]: these are the pre-trained models. We added four convolution layers with flatten layer and the dense layer (1000). And for all the transfer learning of the model, the prediction is done for dense layers. And the model is compiled for the categorical crossentropy loss, using adam optimizer and giving the accuracy for the metrices.
3) Visualizing the model using visual Keras for knowing the layers present in the particular model.
4) In data augmentation preparing the image data generator for rescaling the image to 1. /255, rotation angle =40, width and height shaft range, shear range, and zoom range = 0.2 these are the data augmentations done for train datagen, and only rescaling is done for test datagen. And batch size given for all the models is 32.
5) Using the callback functions the models are called for saving the best possible validation accuracy of the model. The checkpoint, early stopping is used for the

validation loss by giving the patience of with minimum mode and minimum delta=0.01 with verbose=1 and log CSV are used to save the logs of the particular model.

6) And now fitting the model with fit generator by giving the training set and validation data, in this the validation data is used as a test set of all the classifier models, and training the model is done for 25 epochs for all the mentioned model in the report.
7) Plotting the loss and accuracy curve for all the models. And evaluated the model for knowing the test accuracy with the loss of the model and train accuracy with the loss of the model.
8) Confusion metrics is plotted for knowing the precision. Recall and F1-score and true-positive rates, true-negative rates, false-positive rates and false-negative rates are calculated using their mathematical formulas and Accuracy is calculated and plotted in the table 1,2,3.
9) Final step is used for detection of pneumonia images by loading the saved models of the trained classifier models and loaded the given input image from the dataset for testing the whether the input image is pneumonia or not.

Graphical representation of the models for Accuracy, loss, Visualization of the layers of the model and Confusion matrix.

CNN model: Consisting of four convolution layers with flatten layer and added the dropout (0.25) overall for the convolution layers and consist of four dense layers (256,128,64,2).

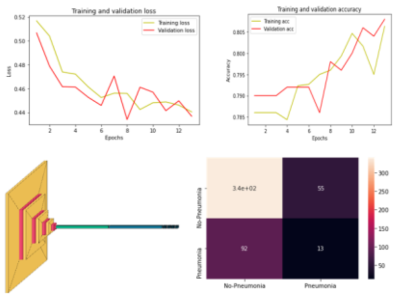


Figure 1: Plotting the loss and Accuracy of the model, Visualizing the layers in the CNN model and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

CNN model [Altered layers]: consist of three convolution layer with one drop out(0.25) and a dense layer of (128, 64,10).

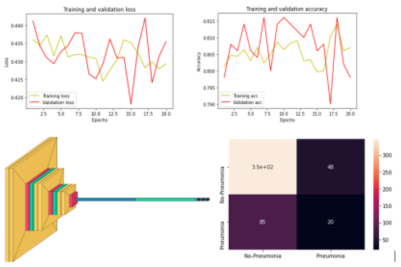


Figure 2: Plotting the loss and Accuracy of the model, Visualizing the layers in the CNN model with Altered Layers and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

ResNet50 Model: The extra flatten layer is used with the dense layer (1000).

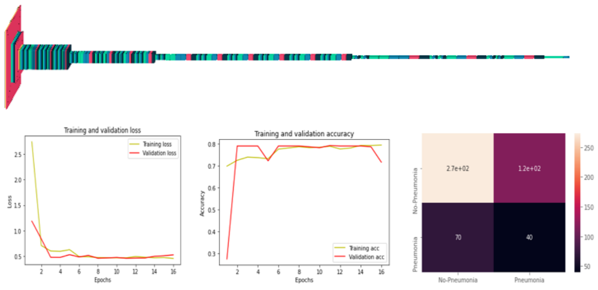


Figure 3: Plotting the loss and Accuracy of the model, Visualizing the layers in the ResNet50 model and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

ResNet50 Model [altered layers]: these are the pre-trained models. We added four convolution layers with flatten layer and the dense layer (1000).

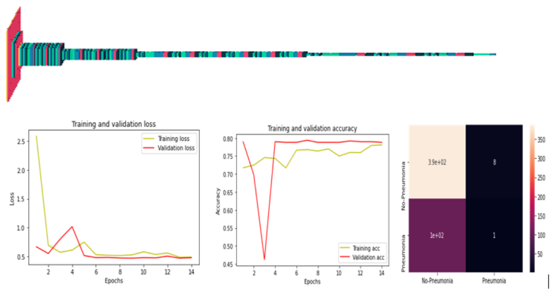


Figure 4: Plotting the loss and Accuracy of the model, Visualizing the layers in the ResNet50 Altered layer model and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

VGG16 model: These are the pre-trained models. We added four convolution layers with flatten layer and the dense layer (1000).

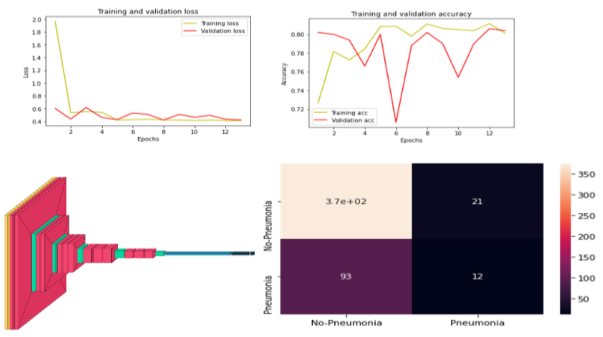


Figure 5: Plotting the loss and Accuracy of the model, Visualizing the layers in the VGG16 model and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

DenseNet121 Model: The model is trained with extra layer of dense layer (1000) with dropout 0.25.

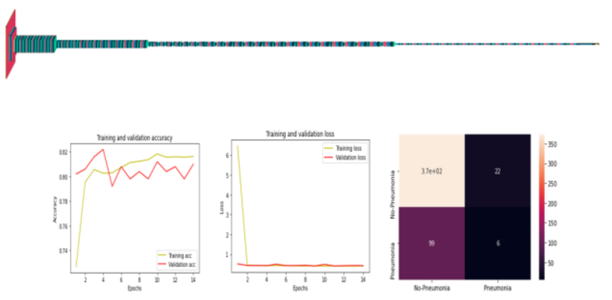
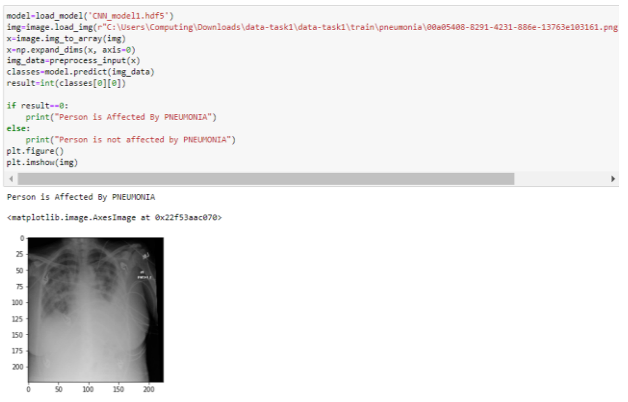


Figure 6: Plotting the loss and Accuracy of the model, Visualizing the layers in the DenseNet121 model and Evaluation metrics of confusion matrix consist of true-positive, true-negative, false-positive and false-negative values.

PREDICTION OF MODEL



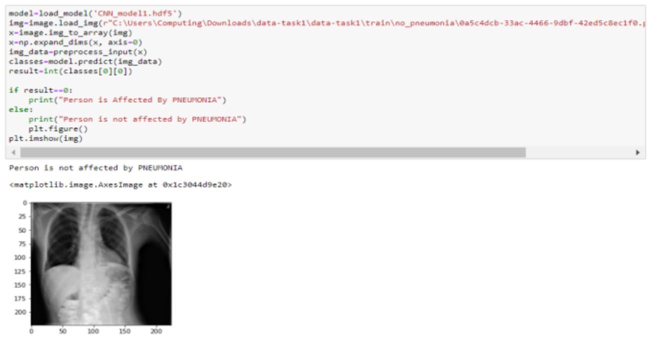
```
model=load_model('CNN_model1.hdf5')
img=image.load_img(r"C:\Users\Computing\Downloads\data-task1\data-task1\train\pneumonia\00a05408-8291-4231-886e-13763e103161.png
x=image.img_to_array(img)
x=np.expand_dims(x, axis=0)
img_data=preprocess_input(x)
classes=model.predict(img_data)
result=int(classes[0][0])

if result==0:
    print("Person is Affected By PNEUMONIA")
else:
    print("Person is not affected by PNEUMONIA")
plt.figure()
plt.imshow(img)
```

```
Person is Affected By PNEUMONIA

<matplotlib.image.AxesImage at 0x22f53aac070>
```

Figure 7: prediction of the model in which the person is affected by pneumonia.



```
model=load_model('CNN_model1.hdf5')
img=image.load_img(r"C:\Users\Computing\Downloads\data-task1\data-task1\train\no_pneumonia\0a5c4dcb-33ac-4466-9dbf-42ed5c8ec1f0.p
x=image.img_to_array(img)
x=np.expand_dims(x, axis=0)
img_data=preprocess_input(x)
classes=model.predict(img_data)
result=int(classes[0][0])

if result==0:
    print("Person is Affected By PNEUMONIA")
else:
    print("Person is not affected by PNEUMONIA")
    plt.figure()
plt.imshow(img)
```

```
Person is not affected by PNEUMONIA

<matplotlib.image.AxesImage at 0x1c3044d9e20>
```

Figure 8: prediction of the model in which the person is not affected by pneumonia.

Evaluation formulas

The formulas for calculating the precision, recall and f1score are:

precision = TP / (TP + FP) (should be multiplied by 100)
recall = TP / (TP + FN) (should be multiplied by 100)
f1 = 2 * precision * recall / (precision + recall)

The formulas for finding the true-positive rates, true-negative rates, false-positive rates and false negative rates are:

TPR = TP / (TP + FN)
TNR = TN / (TP + TN)
FPR = FP / (TP + FN)
FNR = FN / (FP + TN)

Model	Train Accuracy	Training loss	Test Accuracy	Testing loss	Test Metrics 'Accuracy'
CNN Model	79.83%	0.44	80.80%	0.44	70.6%
CNN Model [adding more layers]	81%	0.42	79.80%	0.44	73.4%
DenseNet121 Model	81.80%	0.39	81%	0.43	75.8%
VGG16 Model	80.60%	0.42	80.40%	0.42	77.2%
ResNet50 Model	75.405	0.50	71.60%	0.53	60.8%
ResNet50 Model [Adding more layers]	78.73%	0.47	78.80%	0.48	77.60%

MODEL	Labels	Precision (should be multiplied by 100 for percentage)	Recall (multiply by 100 for percentage)	F1- score
CNN Model	0 1	0.79 0.19	0.82 0.15	0.82 0.15
CNN Model [adding more layers]	0 1	0.80 0.29	0.88 0.19	0.84 0.23
DenseNet121 Model	0 1	0.79 0.29	0.94 0.06	0.86 0.09
VGG16 Model	0 1	0.80 0.36	0.95 0.11	0.87 0.17
ResNet50 Model	1	0.25	0.42	31
ResNet50 Model [Adding more layers]	0 1	0.79 0.11	0.98 0.01	0.87 0.02

MODEL	TPR	TNR	FPR	FNR
CNN Model	0.12380952380952381	0.9631728045325779	0.5238095238095238	0.23291139240506328
CNN Model [altered layers]	0.19047619047619047	0.9455040871934605	0.45714285714285713	0.21518987341772153
DenseNet121 Model	0.05714285714285714	0.9841688654353562	0.20952380952380953	0.25063291139240507
VGG16 Model	0.11428571428571428	0.9689119170984456	0.2	0.23544303797468355
ResNet50 Model [Altered layers]	0.009523809523809525	0.9974226804123711	0.0761904761904762	0.26329113924050634

Table: Evaluation results of the models.

RESULTS

From the above values which are plotted in the tables. Overall, when the models are compared with each other. In terms of training accuracy, it is seen that the train accuracy of DenseNet121 model is more (81%) with that of its testing accuracy (81%). Even though the ResNet50 Model metrics accuracy is low (60.8%) but the confusion matrix of this model values for precision (25%) and Recall (42%) and yielded good F1-score of 31 when it is compared with all the other models. Overall, for the VGG16 model it achieved high precision (36%) when it is compared with all the models.

CONCLUSION

For pneumonia detection in medical images using the dataset provided in the Assessment, we have used the Convolution neural networks (CNN model, CNN model with altered layers) and transfer learning techniques like DenseNet121, VGG16, ResNet50, and ResNet50 with altered layers. During preprocessing layers, adding the layers or removing an extra layer so that the model should not be overfitted and in such a way the model is tweaked in that part by adding the dropout and dense layers etc. and predicting the pneumonia is there or not by giving the input image from their paths. Overall VGG16model achieved high precision (36%) than the DenseNet121 and CNN altered layered models with (29%) and when we considered F1-score the ResNet50 model achieved high f1-score of 31 and precision (25%) and recall of (42%). For giving the better performance the TPR, TNR should be high when it is compared with FPR and FNR. From our model it is seen that TNR is high when it is compared with FNR and TPR of VGG16model and slightly attains the TPR when it is compared with its FPR. In future perspective as the provided data is imbalanced the preprocessing model layers can be further tweaked to attain more Accuracy and High precision-recall values.

Task 2: Game Learning for gym-super-mario-bros.

***Abstract*: In this assessment, we are tackling the problem of game learning i.e., gym-super-Mario-bros by using the deep reinforcement learning agents which receive the image inputs from the game-simulator and that self plays the game autonomously. The deep reinforcement learning agents that are considered in this assessment are PPO, DQN, and A2C agents were used for self-playing the super Mario bros game using the stable- baselines for training and evaluating the agents by hyperparameter tuning and the agents were trained for 1 million total time steps. And also trained the PPO agent without adding the baselines for 3000 episodes. The performance of the agents has been noted by training the agents with three different seeds and evaluating the agents using the metrics such as Average reward, Average game score, Average steps per episode, Average Q-value, training loss, and the training time and testing time is taken for each agent is noted in the report.**

Keywords: Proximal policy-optimization (PPO), Deep Q-Network (DQN), Advantage Actor-Critic (A2C), gym super-mario-bros, agents, policies.

INTRODUCTION

Reinforcement learning: is the feedback-based ML technique in which the agent tries to learn for behaving in the environment by performing the actions and also seeing the result of that action. The agent learns by its experience since there is no labeled data. Agents are used to exploring the environment and acting upon it. And policy a kind of strategy which is applied by the agent for taking the next action which is based on its current state. Agents interact with the environment through the sequence of actions, rewards, and observation [6]. The main focus of reinforcement learning is maximizing the reward signal by learning with its own experience.

Deep Q Network: Deep Q-networks (DQN) is the value-based reinforcement learning and Proximal policy optimization (PPO) is the policy-based reinforcement learning [5]. DQN agents use the exploration policies such as Softmax [5].

A DQN agent chooses the actions based upon $\pi * \theta$ (st) = arg max $Q\theta$(st , at), where $Q* \theta$ (s, a) = maxπ E[rt + γrt+1 + γ 2 rt+2 + ...|st = s, at = a, π, θ] with weights θ and which are learnt by the CNN or MLP neural network [5]. It consist hyperparameters such as batch size, discount factor, learning rate, initial exploration and final exploration [5].

Proximal Policy Optimization:

Proximal Policy Optimization: Proximal policy optimization (PPO) is used for optimizing the policies by alternating the sampling data and performed with several epochs of the optimization. PPO is simpler to implement and has some benefits over the region policy optimization [7]. PPO is a policy gradient approach that performs each policy update using several stochastic gradient ascents. The goal of the PPO is to provide a policy gradient estimator that can be used in a stochastic gradient ascent. To do this, significant policy modifications in the policy gradient were avoided.

Random seeds are used for creating the training and test dataset because the main goal is that make sure of getting the same training and validation dataset when using different parameters for assessing the performance of different models. Seeds are the initialization state for the pseudo number generator. Training curves track the agent's average score with the average predicted action values [6].

Advantage Actor Critic: A2C (Advantage Actor-Critic) is the combination of two types of reinforcement learning algorithms they are policy-based and value-based algorithms. In PPO, and in this optimization, the Actor is used to sample the actions in the state, and critics are used for estimating the value in the state [5].

Super Mario Bros Environment: It is the game available from OpenAI Gym, it is a two dimensional platform game which requires the player for control the character named mario who completes the game by beating the levels. But here the agents are trained to self-play the game in the environment learning through the training of the model through the episodes or through the time taken steps by learning through all the levels by killing the enemies and jumping over the obstacle and preventing from losing its lives.

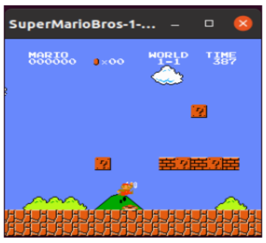


Figure 1: Gym super mario game environment.

Stable baselines Framework: Based on OpenAI baselines it consist of set of improved implementations of reinforcement learning algorithms. It consists of set of algorithms through which the game is learned by implementing the deep reinforcement learning agents in the model by training and they are A2C, PPO, DQN, GAIL etc. In which each algorithm contains different hyperparameters for tuning the model.

IMPLEMENTATION STEPS

<u>For deep-reinforcement learning agents (for PPO. A2C and DQN) with adding the stable baselines3:</u>

1) Setup the mario game and importing the game and setup the environment ('SuperMarioBros-v0')

2) Preprocessing the environment by installing the pytorch and also installing the stable baselines3.

3) Importing the gym wrappers and gray scale observation and from the stable baselines importing the VecFrameStack and DummyVecEnv for training the game.

4) Now, training the reinforcement learning model from the stable baselines and importing the deep reinforcement learning agents they are used for training the model are: PPO, DQN and A2C. and also importing the base callback for check frequency and saving the path.

SADANAND VITHAL GIRADDI
ID – 25895786
Advanced Machine Learning Assessment

5) Giving the checkpoint directory for the training and saving the log files and then the callback functions are used to save the best model obtained from the training.

6) For all the models trained using the deep reinforcement learning agents are used for "CnnPolicy" only. At first the models are tested without giving the seed and later the model is tested for two different seeds for all the deep reinforcement models and the seeds are: 30 and 60.

And for PPO and A2C the hyper parameters are almost same they are: model trained for Cnnpolicy, learning rate=0.000001 and n-steps=512 is taken.

And for DQN agent the model is trained with buffer size=32 as considered as the parameter with Cnnpolicy. It doesnot have n-steps parameter.

5) All the deep reinforcement models are trained with total time steps = 1000000 (10 lakh steps).

6) After completion of training the model is saved and tested it out. With the help of the logs generated from the training of the model the graphs are evaluated for the following metrics such as Average reward, episode reward mean, loss and value loss obtained at each output steps, Average steps taken per episodes and training time and test times is noted for different seeds (30 and 60 are considered during the training for the agents) given to model.

For deep-reinforcement learning agent (for PPO) without adding baselines:

1) Setup the mario game and importing the game libraries. and setup the environment.

2) The seed numbers are given for the environment, action space. and the seed values (32, 64, 128) taken for training the PPO agent without adding any baselines are 32, 64 and 128.

3) the model consists of three convolution layers with different kernel size of (8, 4, 3) and activation ReLU function is considered. And flatten layer is added and linear the model for both the Actor model layer and Critic model layers. the Actor is used to sample the actions in the state, and critics are used for estimating the value in the state [5].

4) The model is solved using the PPO agent solver. And For 10 episodes the checkpoints of the model got saved and the average mean rewards are plotted. And the model does not have the limit it is kept on training until we pressed the stop button.

5) And at every 10 episode the average rewards are evaluated and checkpoints of the model got saved by plotting the graph for average mean reward with the episodes.

EVALUATIONS

Evaluated the values obtained for the deep reinforcement learning agents (PPO, A2C and DQN) using the stable baselines framework. Are episode length mean, episode reward mean, loss, and value loss. And for the deep reinforcement learning for PPO agent, the average reward values are evaluated with the per 10 episodes. Overall, The PPO agent (without-baselines) are trained for 3000 episodes. *All the deep reinforcement learning agents (PPO, DQN and A2C) with stable baselines, the graphs are provided in the Appendix.

Model	Number of Seed	Total time steps taken (1000000)	Episode-Len-mean	Episode-reward-mean	loss	Value loss
PPO (With Baseline)	Without seed	200000	3.85+03	1.6e+03	137	328
		400000	1.42e+03	1.98e+03	94	265
		600000	1.27e+03	2.07e+03	710	653
		800000	1.16e+03	1.85e+03	165	1e+03
		1000000	1.2e+03	2.03e+03	62.6	108
	Seed=32	200000	5.87e+03	1.6e+03	339	756
		400000	1.02e+03	1.76e+03	68.7	181
		600000	1.14e+03	1.93e+03	76.9	146
		800000	1.32e+03	2.11e+03	43.9	113
		1000000	1.62e+03	2.06e+03	93.6	688
	Seed =60	200000	2.06e+03	486	0.139	0.403
		400000	1.65e+03	520	0.0189	0.1
		600000	2.93+03	1.44e+03	187	397
		800000	1.13e+03	1.92e+03	298	372
		1000000	1.15e+03	2.04e+03	250	477

Model	Number of Seed	Total time steps taken (1000000)	Episode-Len-mean	Episode-reward-mean	Loss	Episodes
DQN (With Baseline)	Without seed	200000	1.05e+03	1.6e+03	1.03	124
		400000	1.09e+03	1.58e+03	1.88	308
		600000	1.16e+03	1.69e+03	0.338	484
		800000	1.45e+03	2e+03	0.251	624
		1000000	1.34e+03	1.72e+03	0.864	772
	Seed=32	200000	1.04e+03	1.5e+03	0.861	124
		400000	1.17e+03	1.56e+03	1.53	304
		600000	1.21e+03	1.48e+03	0.175	476
		800000	1.56e+03	1.74e+03	1.42	604
		1000000	1.77e+03	1.93e+03	1.9	716
	Seed =60	200000	2.22e+03	1.65e+03	0.39	92
		400000	1.43e+03	1.62e+03	0.329	228
		600000	1.39e+03	1.7e+03	1.49	376
		800000	1.48e+03	1.88e+03	2.19	512
		1000000	1.3e+03	1.79+03	0.321	668

Model	Number of Seed	Total time steps taken (1000000)	Episode-Len-mean	Episode-reward-mean	Policy loss	Value loss
A2C (With Baseline)	Without seed	200000	1.53e+04	567	-0.224	0.2
		400000	1.71e+04	573	-0.00403	0.201
		600000	1.81e+04	604	162	9.28e+03
		800000	1.81e+04	613	-0.336	0.174
		1000000	1.85e+04	607	0.205	0.381
	Seed=32	200000	1.77e+04	511	0.236	0.616
		400000	1.8e+04	543	0.0568	0.43
		600000	1.82e+04	569	-0.00744	0.705
		800000	1.79e+04	574	-0.00165	0.416
		1000000	1.84e+04	574	-0.162	0.164
	Seed =60	200000	1.74e+04	554	-0.0767	0.323
		400000	1.86e+04	535	36.6	1.05e+03
		600000	1.83e+04	566	0.198	0.418
		800000	1.81e+04	579	-0.106	0.203
		1000000	1.82e+04	577	0.172	0.188

Model	Number of Seed	Average Reward	Episode-
PPO (without baseline)	Seed=32	902.700	500
		1045.09	1000
		1287.80	1500
		2066.00	2000
		2621.39	2500
		1152.40	3000
	Seed=64	886.79	500
		1079.30	1000
		1038.5	1500
		1911.80	2000
		1555.0	2500
		2815.10	3000
	Seed =128	952.09	500
		1307.69	1000
		1835.80	1500
		2190.0	2000
		812.0	2500
		1431.30	3000

Table: Evaluation results for the Deep reinforcement Learning agents (PPO, DQN and A2C) with different seeds (30 and 60) by using the stable baselines. And deep reinforcement learning of PPO agent with different seed (32,64,128) without using the baselines.

The deep reinforcement learning for (PPO, DQN, A2C) agent with baselines: In PPO model for each seeds the learning rate=0.000001 and n step is given as 512. So, it says that for getting one output values it takes 512 steps and predicted the evaluation for Episode length mean, Episode reward mean, loss, value loss. Whereas in DQN model, here it doesn't have n step parameter instead of that the buffer size =32 is given and it starts with taking large steps i.e., nearly 50,000 to 60,000 steps for predicting the 1st output value and from the 2nd step it takes nearly 6000-7000 steps for predicting the values for 2nd output. In this model it predicted the values for Episode length mean, Episode reward mean, loss values with each episode. And A2C model, even though we have given the n steps as 512, but it is seen that for every steps it takes 51200 steps for giving the 1st output where the n steps are multiplied by 100 in the A2C model and for predicting the rest of the values also it takes the same number of steps i.e., 51200 steps. In this model it predicted the values for Episode length mean, Episode reward mean, policy loss and value loss. Overall, In PPO model when the loss is taken into consideration from the above table the loss is decreased with the seed number of 32 when it is compared with the other two seed of the same model with increasing in the total time steps taken and episodes mean reward are increased for all the seeds given in the PPO model with the total time steps taken.

In DQN model, the episode length mean values are increased with the episodes over total time of steps taken for all the seeds given to the model. And the loss of the model goes on decreasing with the total time steps taken and episodes are considered for the DQN model without seed.

In A2C model, the episode length mean values and episode reward mean values are increased with the episodes over total time of steps taken for all the seeds given to the model.

The deep reinforcement learning for PPO agent without baselines: when PPO agent is trained with different seeds of 32, 64 and 128 for 3000 episodes for each model. From the evaluation results it seen that PPO agent with seed 64 gives highest average reward of 2815.10 for 3000 episodes with mario score of 1300 when it is compared with the same PPO agent with 32 and 128 seeds. As when we give more seed (128) it is observed that mario tries to achieve high score and achieved high score of 1400 and also observed that mario plays more faster and gets out immediately by achieving the high score. And when the PPO agent with 32 seeds gives good average reward of 1152.40 for 3000 episodes and the mario was learning through the episodes slowly but clearly and taking longer time and staying in the environment for a long time and scoring the scores, but it is also seen that PPO agent with high seed of (128) gives medium average reward even though the mario score is high. For 2000 episodes, the average reward of an agent with 128 seed number was having highest reward of 2190.

The highest scores of the mario given by learning through its experience for 3000 episodes are:

Highest mario score for PPO agent with (32 seed and 64 seed):

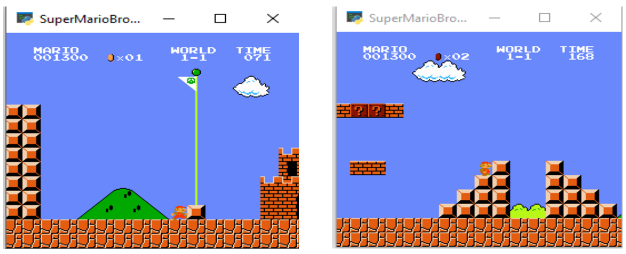


Figure 2: super mario highest score is 1300 for 3000 episodes.

Highest mario score for PPO agent with (128 seed):

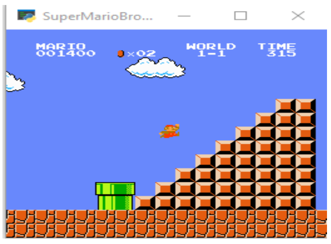
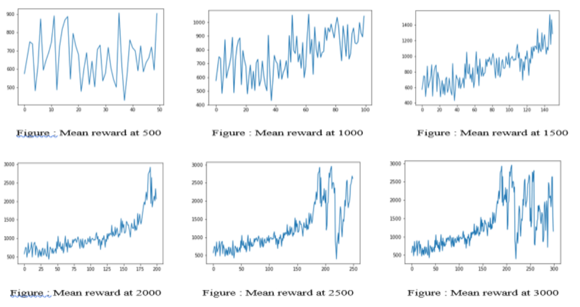
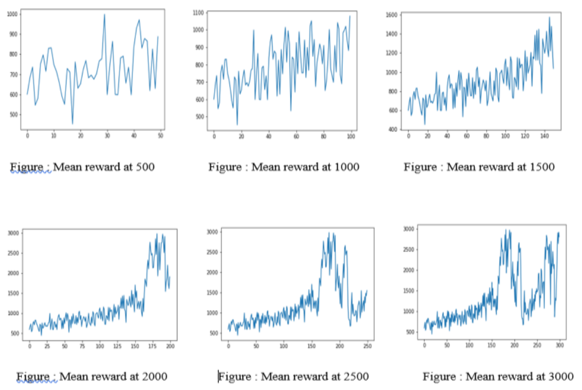


Figure 3: super mario highest score is 1400 for 3000 episodes.

Graphs obtained for PPO agents for model without adding any baselines for seed =32.



Graphs obtained for PPO agents for model without adding any baselines for seed =64.



Graphs obtained for PPO agents for model without adding any baselines for seed =128.

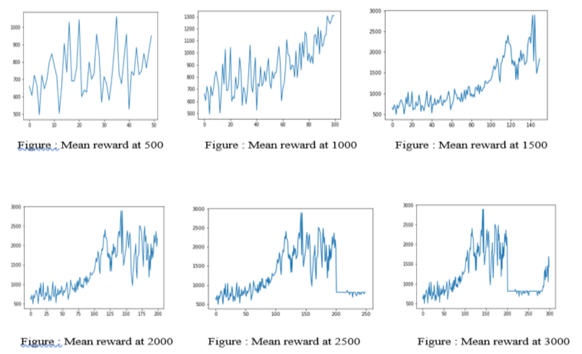


Figure 4: The above graphs represent the Average mean reward vs Episodes (PPO agent with seed (32,64, 128) without baselines).

CONCLUSION

For game learning: The super mario game is trained using the deep reinforcement learning agents they are PPO, DQN and A2C for without adding the seed and adding the two different seed (32 and 60) by adding the stable baselines. And also trained the PPO model for three different seed 32,64 and 128 without adding any baselines. The time taken for training the deep reinforcement learning agents (PPO, DQN and A2C) is for PPO model for training each seed, it took nearly 5 to 6 hours. For DQN model for training each seed, it took nearly 3 to 4 hours. And for A2C model for training each seed, it took nearly 2 to 3 hours. Overall, it is seen that for training the DQN agent with different seeds, when the seed number increased the average number of steps also increased in predicting the output values for episodes. And evaluated the results for episode length mean, episode mean reward, mario game scores, loss, policy loss, value loss for total time steps taken and episodes.

REFERENCE

[1]. Polat, Ö., 2021, August. Detection of Pediatric Pneumonia from X-Ray Images using ResNet50 and GAL Networks. In *2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)* (pp. 1-6). IEEE.

[2]. Convolution Neural Networks, Lecture 3 advanced machine learning, Blackboard University of Lincoln, [Dr. Lei Zhang]

[3]. Varshni, D., Thakral, K., Agarwal, L., Nijhawan, R. and Mittal, A., 2019, February. Pneumonia detection using CNN based feature extraction. In *2019 IEEE international conference on electrical, computer and communication technologies (ICECCT)* (pp. 1-7). IEEE.

[4]. Labhane, G., Pansare, R., Maheshwari, S., Tiwari, R. and Shukla, A., 2020, February. Detection of pediatric pneumonia from chest X-ray images using CNN and transfer learning. In *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)* (pp. 85-92). IEEE.

[5]. Lecture 7, Advanced machine learning, blackboard lincoln, Professor [Heriberto Cuay´ahuitl].

[6]. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *nature*, *518*(7540), pp.529-533.

[7]. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.