Deep Learning

Final Project: Zelda Game Level Classification

7/4/2020

Mohammad Ayyaz Azeem
SAP ID 12520

Contents

1. Introduction: 3
2. Transfer Learning Methods: 3
2.1 Inception V3 3
2.2 ResNet50: 4
2.3 VGG16 4
3. Model selection: 6
4. Model Training: 6
4.1 RMSprop 6
4.1.1 Inception v3 model with RMSprop Optimizer: 6
4.1.2 ResNet50 with RMSprop Optimizer 7
4.1.3 VGG16 with RMSprop Optimizer 8
4.2 ADAM 8
4.2.1 Inception v3 model with Adam Optimizer: 8
4.2.2 ResNet50 with Adam Optimizer 9
4.2.3 VGG16 with Adam Optimizer 10
4.3 Regularization 10
4.3.1 Inception v3 model with Regularization: 10
4.3.2 ResNet50 with Regularization 11
4.4 Custom model: 11
5. Conclusion: 13
6. Future Work 14
7. References 14

1. Introduction:

The convolutional neural network (CNN) is a set of deep learning artificial neural networks widely used in image labeling, object detection, image recognition and classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars as well as from healthcare to security. (Thomas, 2019)

Convolutional Neural Network is state of the art image detection neural networks architecture. The concept is to take an image, convert it into grid of pixel data. Basic CNN structure is as follows: Convolution -> Pooling -> Convolution -> Pooling -> Fully Connected Layer -> Output. Convolution means taking the original data, and creating feature maps from it. Pooling means down-sampling, like "max-pooling," where we select a region, and then take the maximum value in that region and that becomes the new value for the entire region. We select a window for convolution, slide it across the image and perform pooling on the way. Each convolution and pooling step is actually a hidden layer. After this, we have a fully connected layer, followed by the output layer. The fully connected layer is your typical neural network (multilayer perceptron) type of layer, and same with the output layer.

Convolutional Neural Networks (CNNs) are comprised of neurons that self-optimize through learning. Each neuron will receive an input and carries out an operation (such as a scalar product followed by a non-linear function) which is the basis of countless ANNs. The transition from the input raw image vectors to the final output of the class score, the entire network will represent a single perceptive score function (called the weight). The last layer contains the loss functions associated with the classes, and all of the standard operations developed for traditional ANNs still apply. The only prominent difference between CNNs and traditional ANNs is that CNNs are mostly used in the field of pattern recognition within images. This permits us to encode image-specific features into the architecture, making the network. (O'Shea & Nash, 2015)

The deep learning architecture used for transfer learning in our experiments are Inception V3, ResNET50 andVGG16.

2. Transfer Learning Methods:

2.1 Inception V3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2015):

Inception V3 by Google is the 3rd version in a series of Deep Learning Convolutional Architectures. Inception V3 was trained using a dataset of 1,000 classes from the original ImageNet dataset which was trained with over 1 million training images. (Adam, 2019)

Inception network is a state-of-the-art deep learning architecture (or model) for solving image recognition and detection problems. (Inception Layer) is a combination of all those layers (namely, 1×1 Convolutional layer, 3×3 Convolutional layer, 5×5 Convolutional layer) with their output filter banks concatenated into a single output vector forming the input of the next stage.

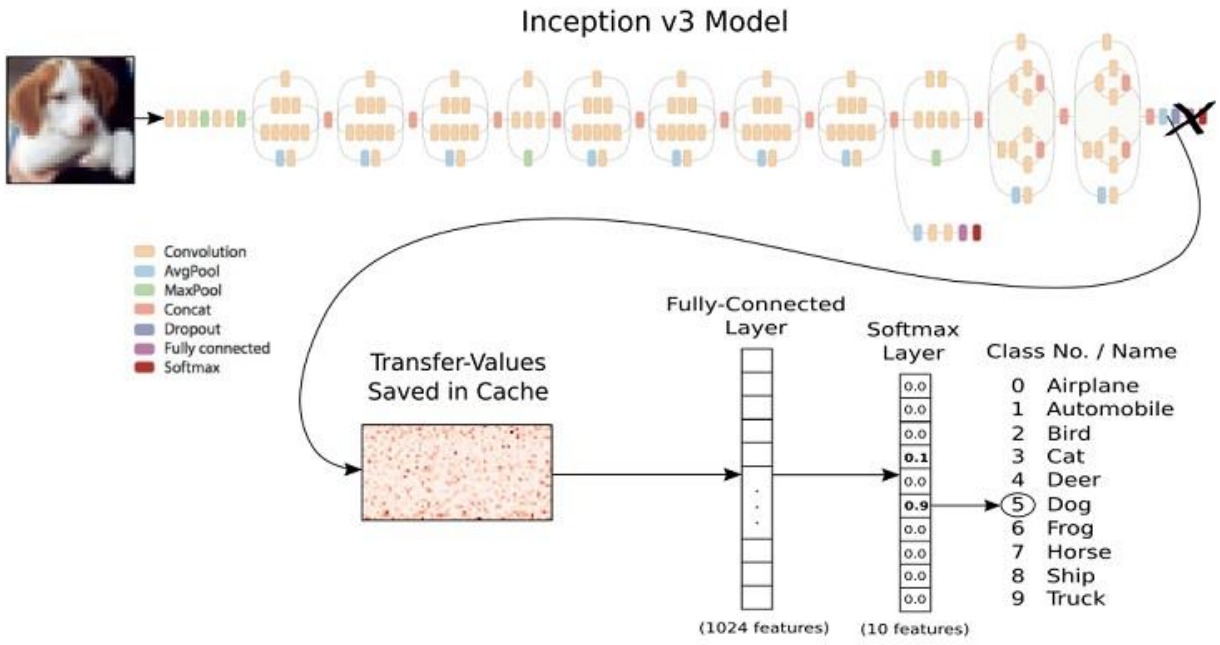

Figure 1 Inception V3

2.2 ResNet50:

ResNet50 is a variant of ResNet model which has 48 Convolution layers along with 1 MaxPool and 1 Average Pool layer. It has 3.8 x 10^9 Floating points operations. It is a widely used ResNet model. We have explored this in depth. (Kaiming He, 2015)

Residual learning framework ResNet is a powerful backbone model that is used very frequently in many computer vision tasks. On the ImageNet dataset residual nets with a depth of up to 152 layers---8x deeper than VGG nets but still having lower complexity.

AlexNet, the winner of ImageNet 2012 and the model that apparently kick started the focus on deep learning had only 8 convolutional layers, the VGG network had 19 and Inception or GoogleNet had 22 layers and ResNet152 had 152 layers. ResNet-50 that is a smaller version of ResNet 152 and frequently used for transfer learning. ResNet uses skip connection to add the output from an earlier layer to a later layer. This helps it mitigate the vanishing gradient problem.

2.3 VGG16 – Convolutional Network for Classification and Detection:

VGG16 is a convolutional neural network model developed by by Simonyan and Zisserman for ILSVRC2014 competition (Zisserman, 2015). The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. VGGNet consists of 16 convolutional layers with only 3x3 kernels.The Architecture depicted below is VGG16.

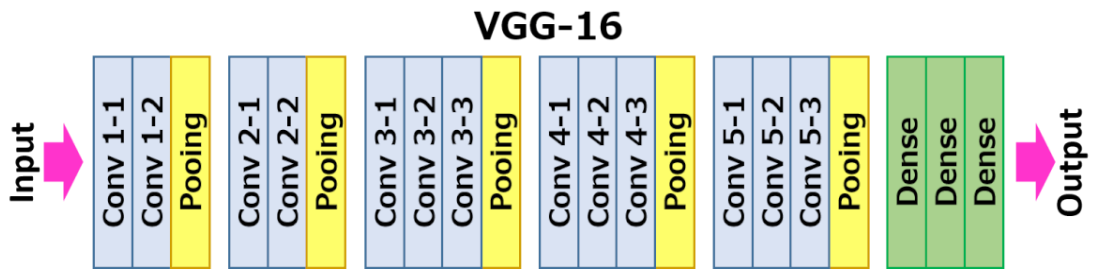

Figure 2: VGG16

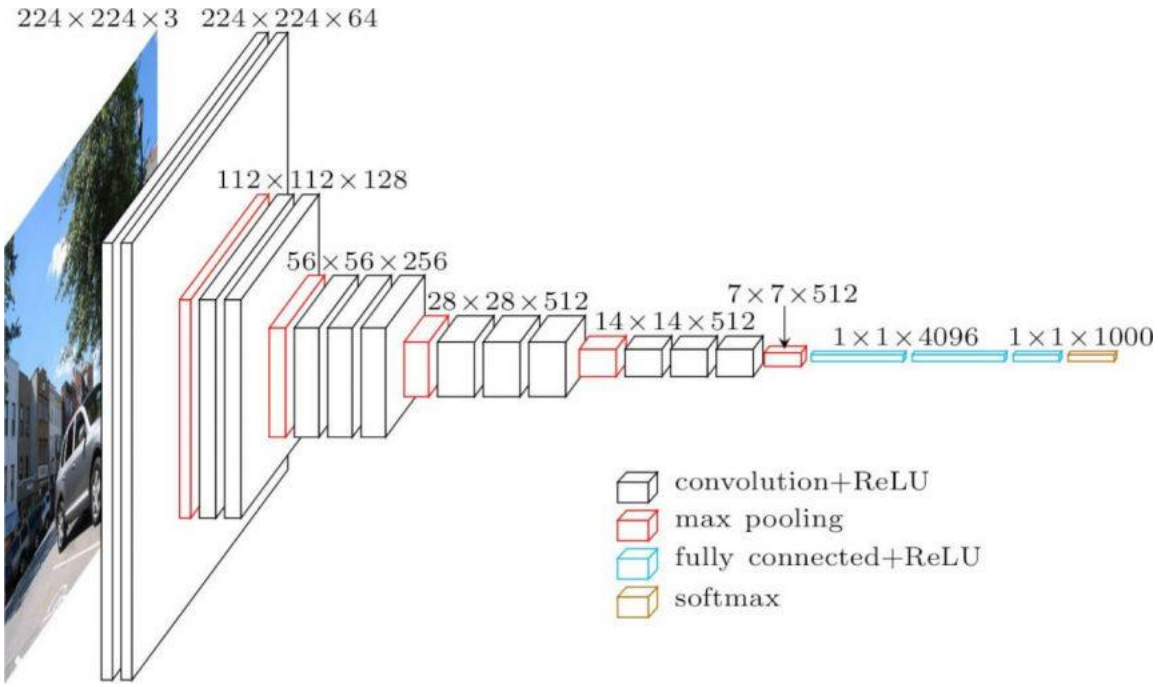

Figure 3: VGG16 internal structure

The input to cov1 layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2. (Hassan, 2018)

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks. (Hassan, 2018)

3. Model selection:

Preprocessing: 15 playable and 15 unplayable images were used as test images for performance evaluation of the models, in total 2018 images were used for training. Colored images were used throughout the experiments.

Data Augmentation: Data augmentation is performed when available training dataset is limited in order to improve the overall model performance. In order to train our model, data augmentation is performed by applying 40 degree random rotation on images along with vertical flip, horizontal flip, random shifts and adding it to the original dataset. This data augmentation step is performed on both training and held-out test datasets.

4. Model Training:

By making use of Transfer learning, the final layer of an existing model can be retrained, ultimately resulting in not only decreasing the training time. We have employed VGG16, Inception V3 and ResNet50. Being able to retrain the final layer means that one can preserve the acquired knowledge that the model had learned during its original training and apply it to one smaller dataset, ultimately resulting in to highly accurate classifications without the need for extensive training and computational power. (Adam, 2019).

I have added global average pooling layer and dropout factor of 0.4. Then model training was performed using transfer learning with epoch size of 10, batch size of 32, steps per epoch is 20 and validation steps is 5. Image size for each transfer learning model is 299x299 for Inception v3, 224x224 for VGG16 and 300x300 for ResNet50. Optimizers used are RMSProp and Adam with loss function categorical_crossentropy.

4.1 RMSprop

4.1.1 Inception v3 model with RMSprop Optimizer:

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.650 at 10th epoch.

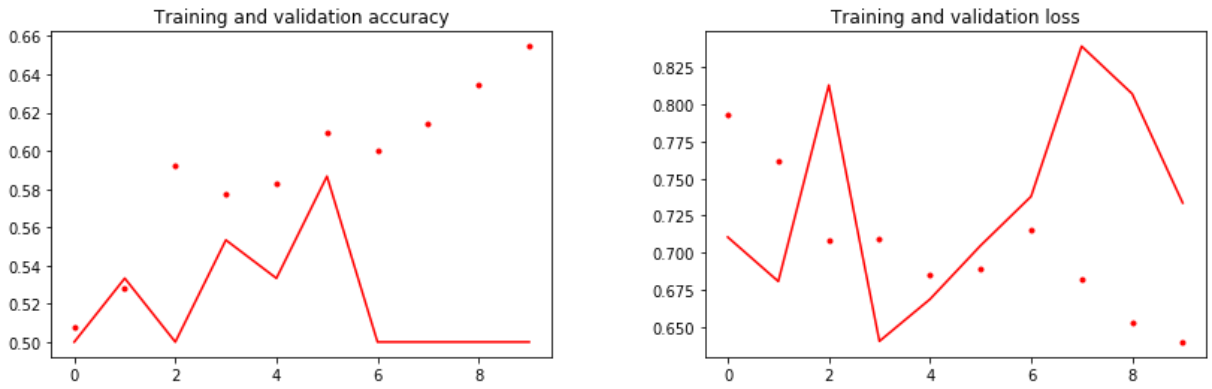

Figure 4 Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.1.2 ResNet50 with RMSprop Optimizer

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.50 at 10th epoch.

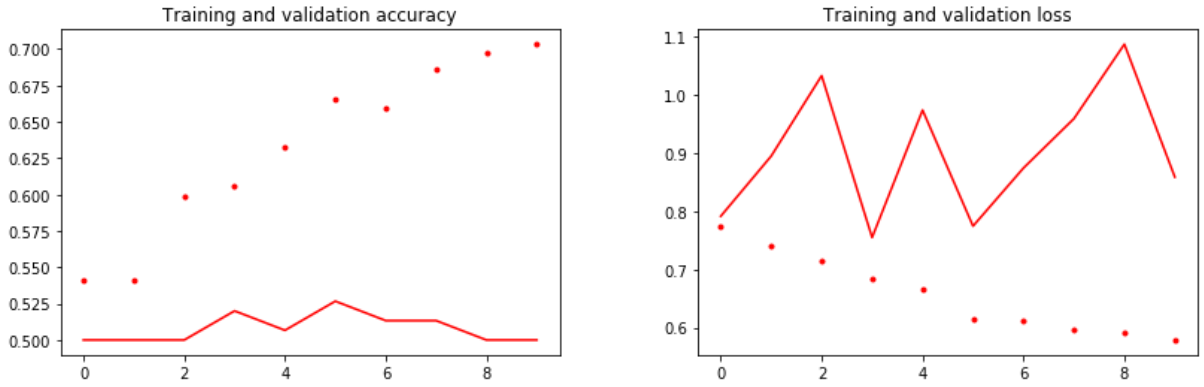

Figure 5: Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.1.3 VGG16 with RMSprop Optimizer

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.580 at 10th epoch.

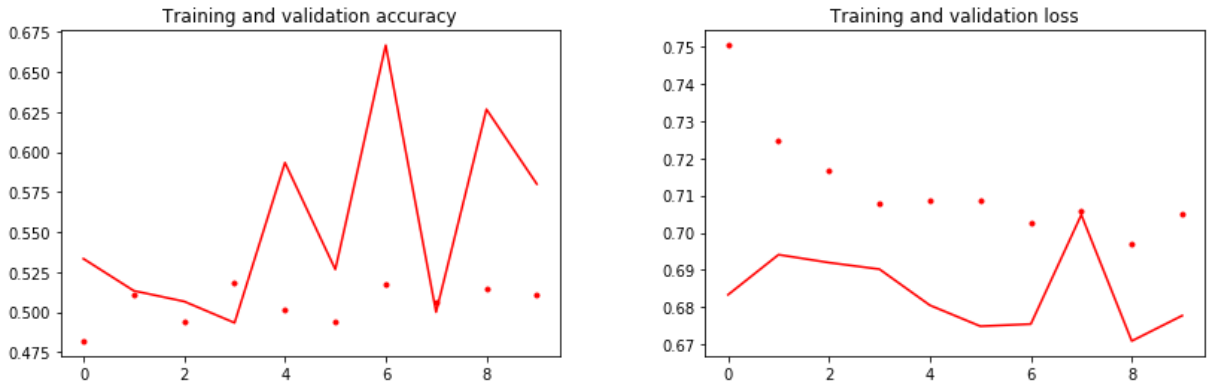

Figure 6: Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.2 ADAM

4.2.1 Inception v3 model with Adam Optimizer:

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.640 at 10th epoch.

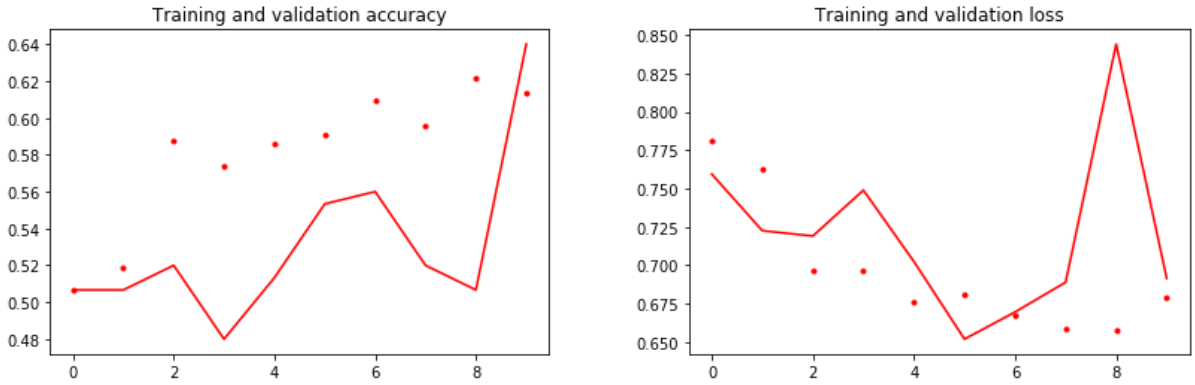

Figure 7 Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.2.2 ResNet50 with Adam Optimizer

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.50 at 10th epoch. As clearly indicated in Figure 8, training data is clearly overfitting (represented by dotted line) so performance on test data set is not good.

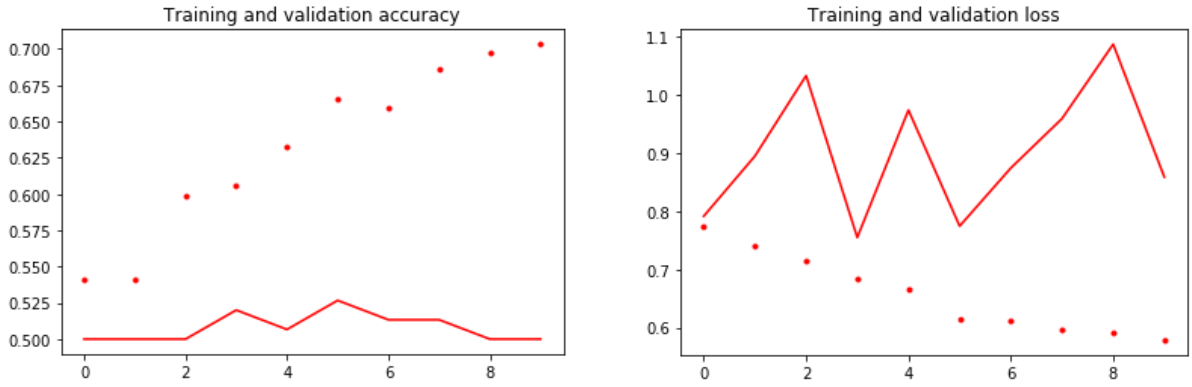

Figure 8: Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.2.3 VGG16 with Adam Optimizer

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.47 at 10th epoch.

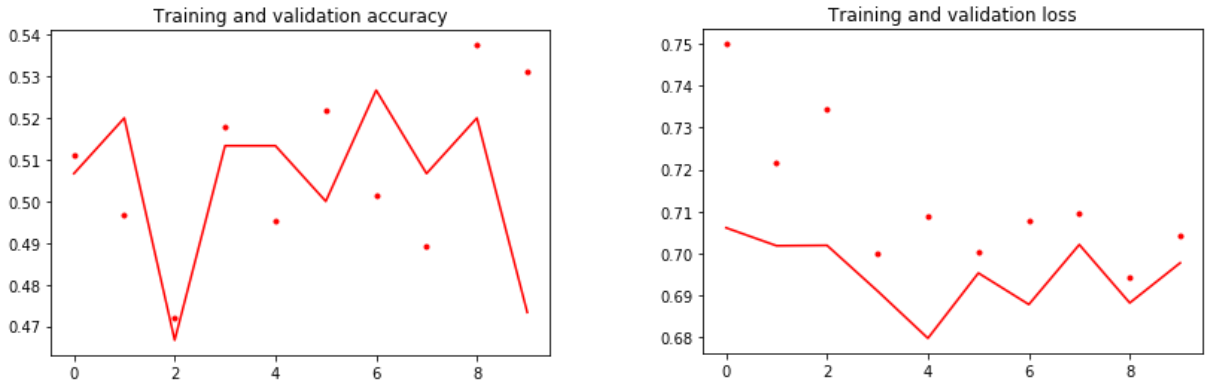

Figure 9: Dotted line represents training data set and straight line represent test data set

The generated confusion matrix results are shown below:

	precision	recall	f1-score	support
zeldaPlayablelevels	0.50	1.00	0.67	15
zeldaUnplayablelevels	0.00	0.00	0.00	15
accuracy			0.50	30
macro avg	0.25	0.50	0.33	30
weighted avg	0.25	0.50	0.33	30

4.3 Regularization

4.3.1 Inception v3 model with Regularization:

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.50 at 10th epoch.

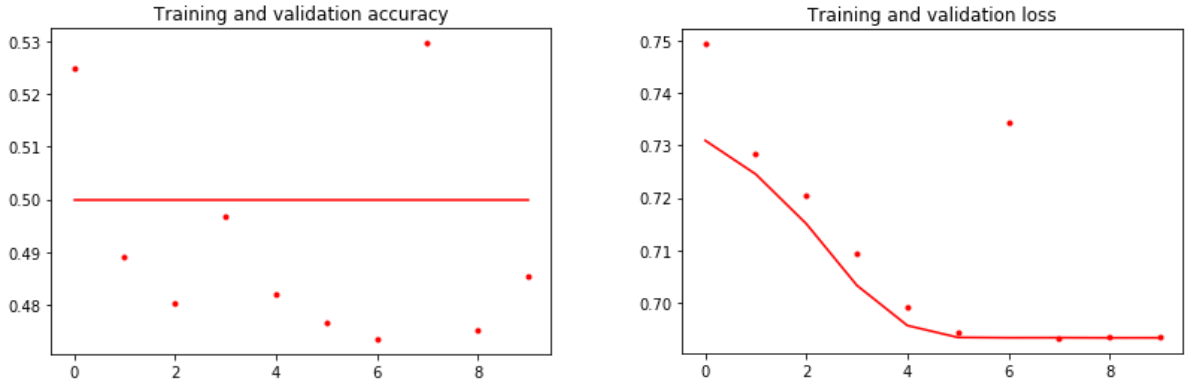

Figure 10: Dotted line represents training data set and straight line represent test data set

4.3.2 ResNet50 with Regularization

Training and validation accuracy and training and validation loss plot are shown below. The maximum validation accuracy achieved is 0.50 at 10^{th} epoch.

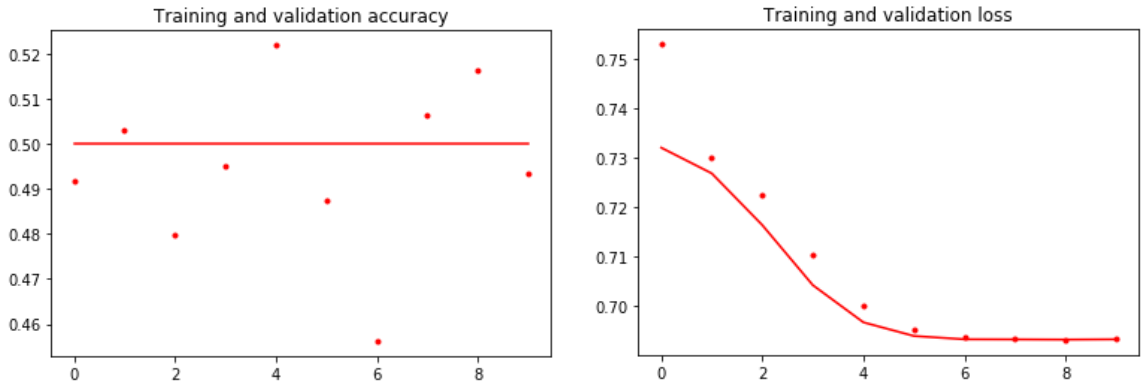

Figure 11: Dotted line represents training data set and straight line represent test data set

4.4 Custom model:

Tensorboard, a tensorflow visualization toolkit, was used to compare the models performance and model selection, the model with the best accuracy on the held out validation set and least loss was selection for further predictions on the test set. Some of the results from Tensorboard are shown below:

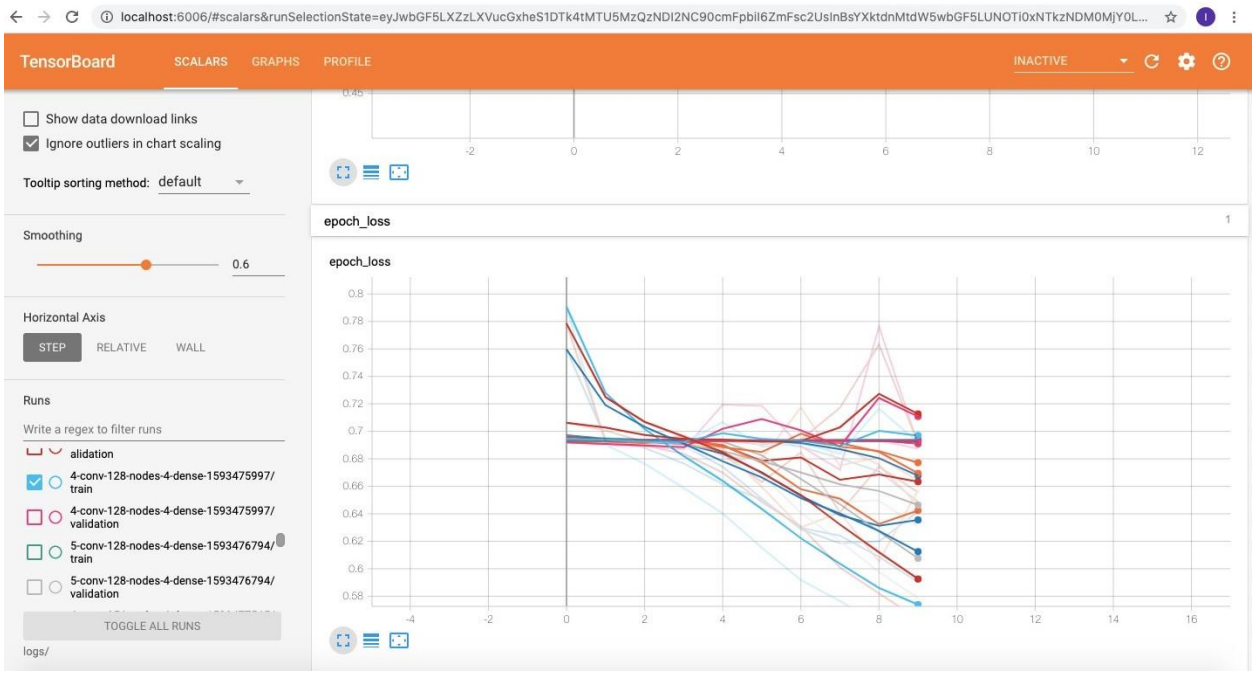

Figure 12 Comparison of Epoch loss during each experiment using different models

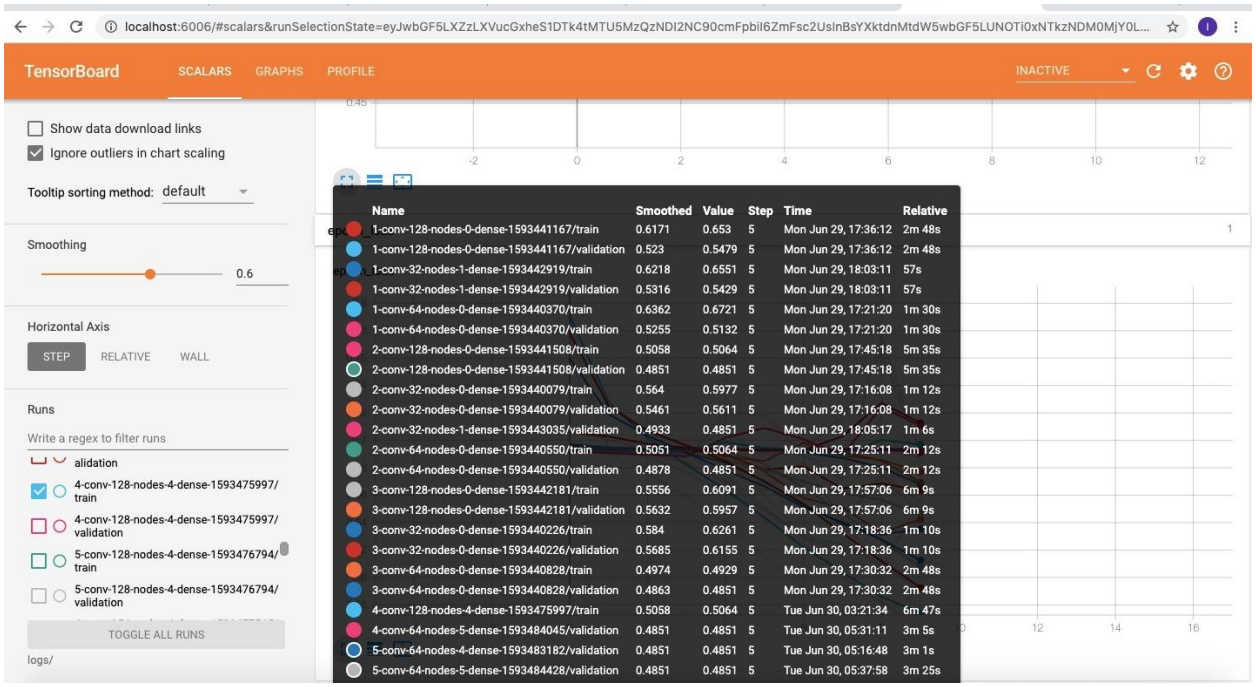

Figure 13 Lists of custom models used for analysis

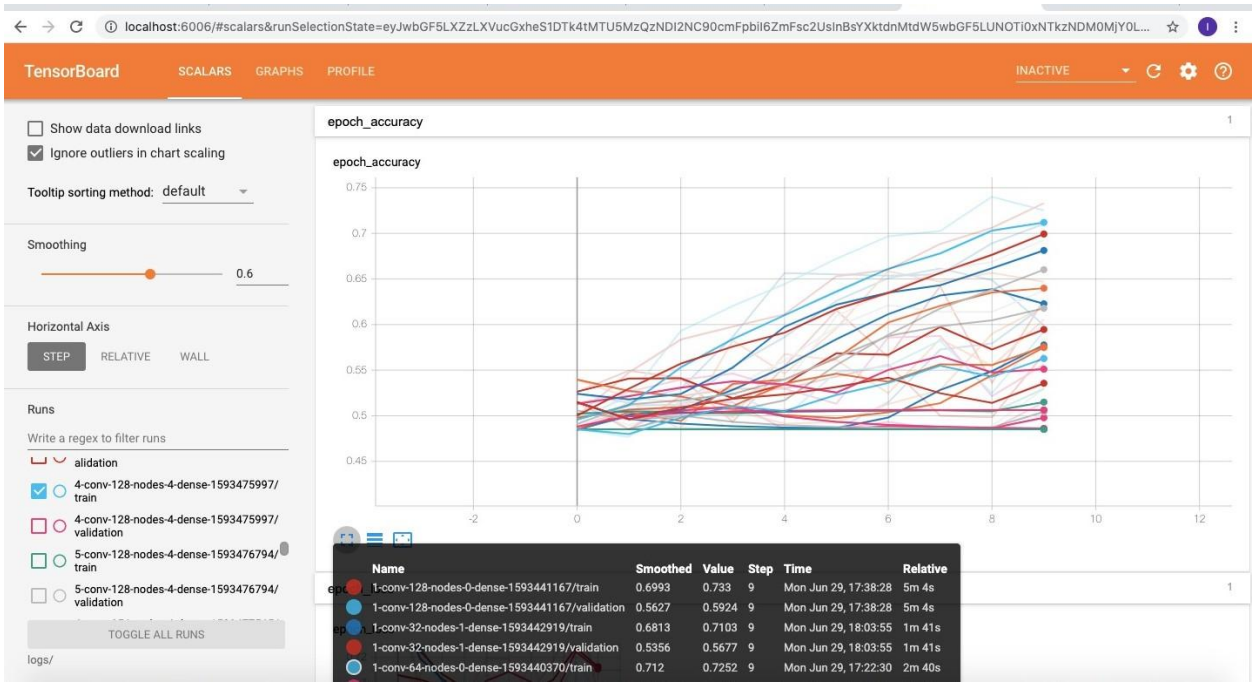

Figure 14 Comparison of epoch accuracy during each experiment using different models

For custom model, different architectures were tested with a combination of dense layers are 0,1,2, 4, 5 of size 16, 32, 64, 128 respectively with convolutional layers of 1, 2, 3, 4, 5. Best model that emerged after comprehensive test is 4-conv-32-nodes-4-dense and the results were as follows: with the validation accuracy 0.80 in epoch 10

```
4-conv-32-nodes-4-dense-1593482505
Model: "sequential 2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 98, 98, 32)        320
_________________________________________________________________
activation 19 (Activation)   (None, 98, 98, 32)        0
_________________________________________________________________
max pooling2d 9 (MaxPooling2 (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 47, 47, 32)        9248
_________________________________________________________________
activation 20 (Activation)   (None, 47, 47, 32)        0
_________________________________________________________________
max pooling2d 10 (MaxPooling (None, 23, 23, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 21, 21, 32)        9248
_________________________________________________________________
activation_21 (Activation)   (None, 21, 21, 32)        0
_________________________________________________________________
max pooling2d 11 (MaxPooling (None, 10, 10, 32)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 8, 8, 32)          9248

activation_22 (Activation)   (None, 8, 8, 32)          0

max_pooling2d_12 (MaxPooling (None, 4, 4, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_10 (Dense)             (None, 32)                16416

activation_23 (Activation)   (None, 32)                0
_________________________________________________________________
dense_11 (Dense)             (None, 32)                1056
_________________________________________________________________
activation_24 (Activation)   (None, 32)                0

dense_12 (Dense)             (None, 32)                1056
_________________________________________________________________
activation_25 (Activation)   (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 32)                1056

activation_26 (Activation)   (None, 32)                0
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 33
_________________________________________________________________
activation 27 (Activation)   (None, 1)                 0
=================================================================
Total params: 47,681
Trainable params: 47,681
Non-trainable params: 0
```

5. Conclusion:

To overcome “limited data” bottleneck problem, we used data augmentation during transfer learning techniques. Furthermore, for model training two optimizers are used Adam and RMSprop with loss function catagorigal_crossrntropy was used; the performance metric indicates that Inception v3 model with Adam Optimizer is appropriate for the dataset and gives better classification model.

As the performance measures indicate that our set of regularization layer did not add value to modeling approach. We have used dense layer with relu activation function, kernel regularizer l2

with learning rate 0.01 and activity regularizer l1 with learning rate 0.01. However, the results were not encouraging as we encountered a flat line indicating no improvement in our experiment results (Figure 10 & Figure 11).

Custom models are used that performs comprehensive analysis and we found best model that emerged is 4-conv-32-nodes-4-dense and the results with the validation accuracy 0.80 in epoch 10.

6. Future Work

The major constraint was limited resources, time and computational power, model training was performed using few epochs. However, results are expected to significantly improve by employing sophisticated CNN models, we can achieve better results by fine-tuning the hyper parameters along the way. CapsuleNet, developed by Geoffrey Hinton & Google in 2017 is an interesting and exciting technique that can be used to perform further analyze the problem set.

7. References

Adam. (2019, 02 17). *AI Developer Program*. Retrieved 07 03, 2020, from Intel: https://software.intel.com/content/www/us/en/develop/articles/inception-v3-deep-convolutional-architecture-for-classifying-acute-myeloidlymphoblastic.html

Hassan, M. u. (2018, 11 20). *VGG16 – Convolutional Network for Classification and Detection*. Retrieved 07 03, 2020, from Neurohive: https://neurohive.io/en/popular-networks/vgg16/

Kaiming He, X. Z. (2015). Deep Residual Learning for Image Recognition. *CoRR* .

O'Shea, K., & Nash, R. (2015). An Introduction to Convolutional Neural Networks.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *CVPR.*

Thomas, C. (2019, 05 27). *An introduction to Convolutional Neural Networks*. Retrieved 07 04, 2020, from Towards data science: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7

Zisserman, K. S. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR.* San Diego, CA, USA.