Transfer Learning for Classification of Pneumonia cases using X-ray Images

Amir Aghdam
{baghalaghdam.a@gmail.com}

1. Introduction

Medicine has been one of the fields that always required high expert knowledge to detect and diagnose diseases that have long crippled human lives. Therefore, in some cases, it becomes very costly to use trained experts for a number of menial tasks, such as detecting abnormalities in medical images. In this regard, computers have made great advancements in extracting useful features from a wide range of images thanks to state-of-the-art algorithms in computer vision and helped doctors to make better decisions.

During recent decades, with the first introduction of Artificial Neural Networks to computer vision by Geoffrey Hinton, Deep Learning techniques obviated the need for hand-engineered features for computer vision tasks. Deep Learning methods are able to learn to extract the most discriminative features from input images without the intervention of humans, and their ability to model every complex function has enabled them the best choice for most of the complex tasks in computer vision.

Despite the impressive performance of the Deep models, they typically require a large set of labeled data for training. Furthermore, it becomes more pressing when dealing with medical datasets in which annotations are costly and require several expert reviews. It means that model might be prone to overfitting or the evaluations of the model performance may not be good estimations of its real performance.

In this work, we propose a Deep learning-based approach for automatic Pneumonia detection from patients' chest images. Pneumonia is an infection that inflames the air sacs in one or both lungs, and it is normally diagnosed through X-ray images of the chest. However, we need to deal with the data imbalance problem while training the model due to the limited number of positive Pneumonia cases compared to normal ones.

We remedy the issues discussed earlier by proposing five major approaches:

- Due to the limited number of data samples of pneumonia cases, we use Synthetic COVID-19 CXR Dataset for better generalization over unseen data.
- To avoid overfitting caused by choosing a large model for a small dataset, we use a lightweight model called Xception.
- Owing to the small number of images in the Dataset, we fine-tune the ImageNet pretrained Xception model on COVID-19 CXR Dataset for better generalization.
- For a more realistic estimate of model performance, we evaluate the model using 5-Fold stratified cross validation.
- To handle the data imbalance problem, we use a weighted Binary Cross Entropy loss and evaluate the model in terms of Precision and Recall as better metrics than accuracy.

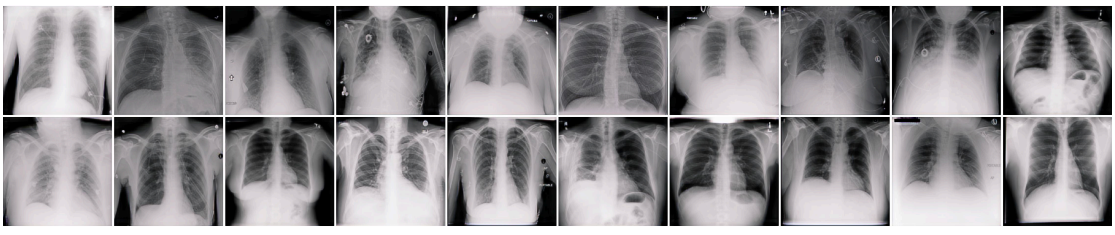

Fig 1. COVID-19 CXR Dataset

2. Approach

2.1 Dataset

Shortage of annotated datasets is a major problem in the medical domain due to its costly nature and privacy issues. As a result, generating medical data using unsupervised methods, such as Generative Adversarial networks (GANs), proves to be an effective solution.

In this work, we use the Synthetic COVID-19 CXR Dataset, which is generated using Conditional GAN, showed in Fig 1. The dataset contains two classes, Normal and Pneumonia, which makes the task a binary classification. It consists of 21,295 images in total, 16,537 images for normal cases, and 4,758 images for pneumonia cases. Each image has a size of (256,256,3). Normal cases account for approximately %77.6 of the images and Pneumonia cases for %22.4 of the whole dataset. Thus, presenting a highly imbalanced dataset.

2.2 Model

In this experiment, we used the Xception model for the Pneumonia classification tasks. This model is built upon the Inception idea to increase the sparsity of the model to empower the model to decide which optimal path to use during training. Inception and Xception both have a similar number of parameters but with a major distinction in their convolutions. Xception is designed using Depth-wise separable convolution with an expansion rate of 1. They have developed an extreme version of Inception by proposing the Xception architecture based on the hypothesis that the mapping of cross-channel correlations and spatial correlations in the feature maps of convolutional neural networks can be entirely decoupled. Xception achieves a state of the art top-5 accuracy of 0.945 on ImageNet. The model architecture is shown in figure 2.

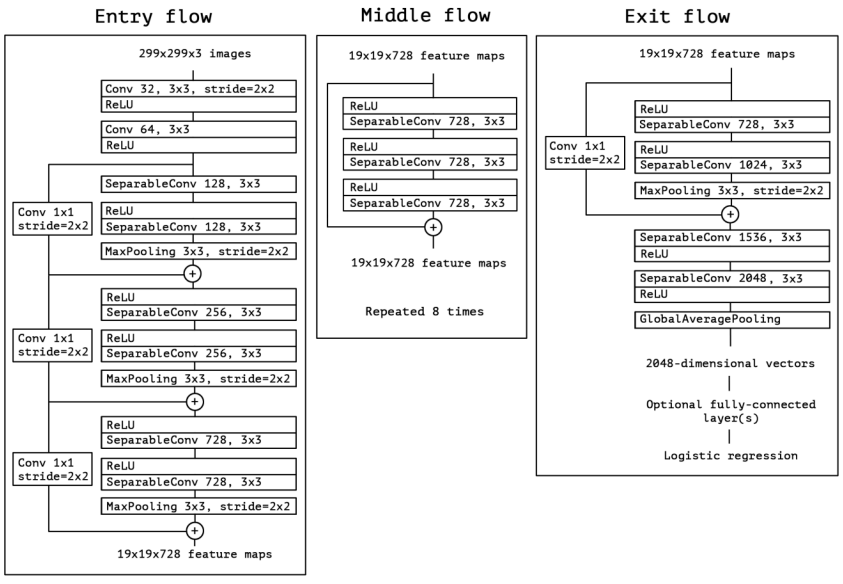

Fig 2. Xception Architecture

2.3.1 Possible Scenarios

Due to the limited number of images, training the model from scratch would most likely lead to overfitting and may not generalize well on unseen cases. In order to address this issue, there are two major solutions:

- A model with a very small number of trainable parameters
- Transferring knowledge from other domains

There are also two possible scenarios for knowledge transfer:

- First, Freezing all layers of the pretrained model, adding a number of trainable layers on top of the model, training those top layers on the dataset then fine tuning all the layers with a very small learning rate.
- Feeding the entire dataset to the network and storing feature maps of a certain intermediate layer of the pretrained network.

2.3.2 Transfer Learning

One issue associated with training a small model is the constrained representation power of the network since the limitation of the number of parameters imposes a limitation on the model design. However, the chosen model, Xception, has a reasonable trade-off between representational power and size.

To offset the issue concerning the limited dataset, we propose to transfer knowledge from other datasets. Transfer learning has a boosting effect on the generalization of the network as there are numerous works affirming this hypothesis.

We hypothesize that knowledge transfer from even the most unrelated domains can also be useful since shallow layers of deep networks are responsible for detecting edges and low-level features. These features could be discriminative enough for deeper layers to extract the semantics with respect to the dataset. That is why we initially train only the deeper layers and froze the rest in training to adapt the pretrained network to our dataset.

Firstly, We add a 2D Global Average Pooling and two Dense Layers, first with 1028 neurons and the last with a single neuron on top of the pretrained Xception Model on ImageNet. Then we freeze the pretrained network and train the added layers on the dataset. After that, we unfreeze the entire layers, except for BatchNorm layers, and fine-tune the entire network with a very low learning rate to adapt the features to our dataset.

2.4 Loss Function

We use the Binary Cross Entropy loss as our classification error metric due to its effectiveness in classification problems. This loss tries to minimize the difference between two distributions of data. However, due to the highly imbalanced samples of the two classes, the model learns to predict the majority class more often since it incurs a lower loss than predicting the minority class.
To handle this issue, we use Weighted Binary Cross Entropy to give a fair misclassification error among two classes. We evaluate our model using 5-Fold Cross Validation and in each training split, we compute the balancing weights using only the training data. In this way, we managed to alleviate the data imbalance.
The Cross-Entropy loss is indicated in Fig. 3, where y^ indicates the predicted label, y true labels, y_i true label of class I, and y^i predicted label of class i.

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

Fig 3. Cross Entropy Loss

3. Experiments and Results

3.1 Train Settings

During the training, we use the input shape of (256, 256, 3) for training with Weighted Binary Cross Entropy. A training batch size of 16 is used because it showed slightly faster convergence than a higher batch size, and we used a 32 batch size for evaluation. We trained the model for 3 epochs and fine-tuned it for 1 epoch to test out our hypothesis. However, a higher number of training epochs would have normally resulted in much better results. Additionally, we keep track of precision, recall, and accuracy during both training and evaluation to monitor the results. Early stopping with the patience of 1 and min_delta of 1e-3 was used along with Model Check Point to save the model with the best validation loss.

3.2 Pre-processing and Augmentation

We zero-centered the input data for both train and evaluation data by dividing all input pixels by 255 and subtracting 0.5, this helps the network for faster convergence as suggested in numerous works.

Random horizontal flip was used as an augmentation technique to aid the network in generalizing well to unseen samples. However, more augmentation approaches can also be used to further improve the results and help the model to learn the distribution of data but settled for the most occurring approach.

3.3 Optimization

We initially used Adam optimizer with a 1e-3 learning rate for training and 1e-5 for fine-tuning. However, after a certain number of epochs, the model gets stuck in a local minimum and fails to improve further.

To handle this, we used a Stochastic Gradient Decent optimizer with a Nesterov momentum of 0.9 and a learning rate decay of 1e-6. An initial learning rate of 1e-3 for training and 1e-5 for fine-tuning worked the best on our dataset.

Fold	Train/Validation Loss	Train/Validation Recall	Train/Validation Precision	Train/Validation Accuracy
1	0.0006 / 0.0018	0.2237 / 0.2256	0.2237 / 0.2256	0.9996 / 0.9995
2	0.0043 / 0.0080	0.2234 / 0.2253	0.2237 / 0.2256	0.9978 / 0.9969
3	0.0022 / 0.0035	0.2236 / 0.2256	0.2237 / 0.2256	0.9986 / 0.9998
4	0.0059 / 0.0066	0.2236 / 0.2256	0.2237 / 0.2256	0.9954 / 0.9972
5	0.0023 / 0.0052	0.2235 / 0.2256	0.2237 / 0.2256	0.9992 / 0.9988
Mean (Validation)	0.0050	0.2255	0.2256	0.9984
Std. (Validation)	0.0021	0.0021	0.0021	0.0011

Table 1. 5-Fold Cross Validation Results

3.4 Evaluation

In order to achieve a realistic estimate of model performance, we used 5-Fold Cross Validation. However, due to the imbalanced data samples, we used stratified 5-Fold Cross Validation to keep the ratio of positive/negative samples uniform in each fold split. Results are reported in table 1. Additionally, a number of train/validation loss and evaluation metrics are reported in Figure 4. The curves for other folds are approximately uniform since the data was stratified in splitting.

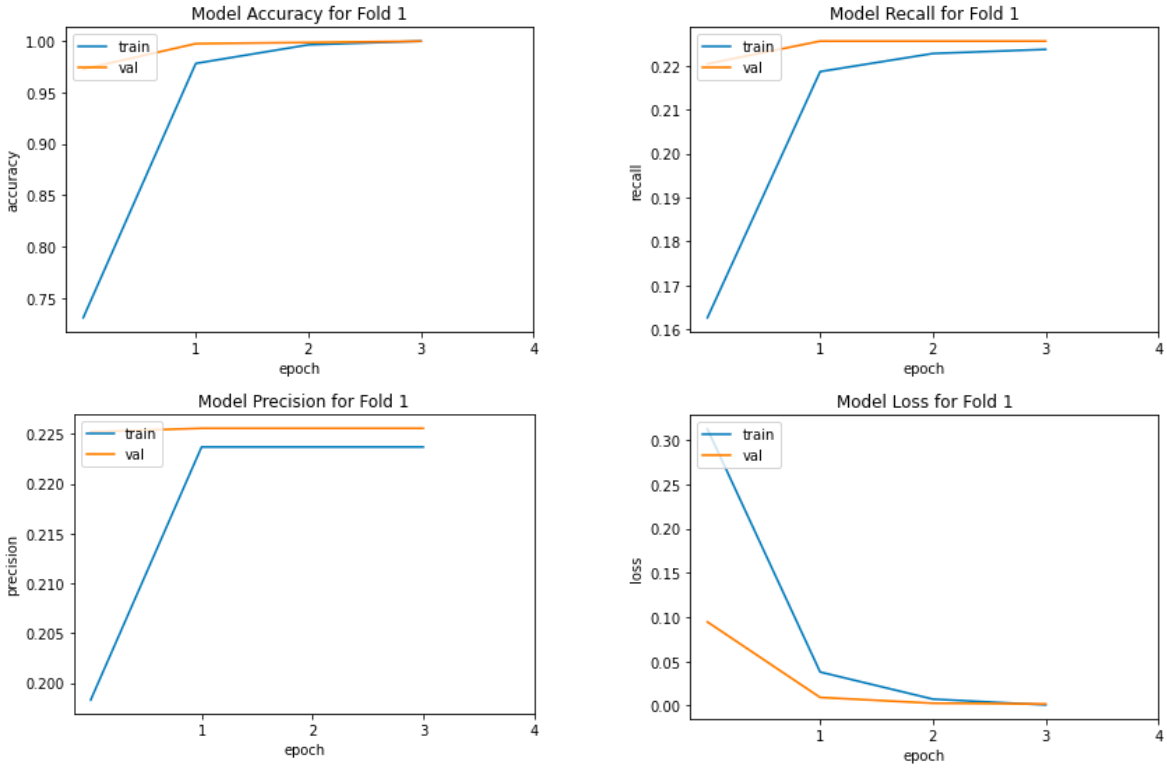

Fig 4. Fold-1 Train/Validation Results

4. Conclusion

In this work, we used the Xception model, which achieved a good trade-off between model size and representation power, for the task of pneumonia classification using chest X-ray images. We proposed a weighted loss to handle class imbalance and transferred knowledge by using pretrained Xception on ImageNet. Additionally, for better estimates of model performance, we evaluate the model using 5-Fold Stratified Cross-Validation.