**Convolution Neural Networks**

**Assignment - 1**

**May 05th , 2020**

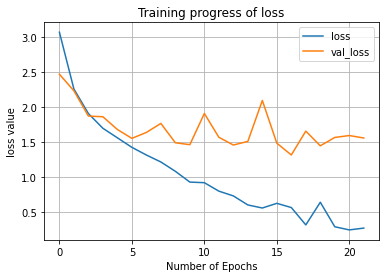
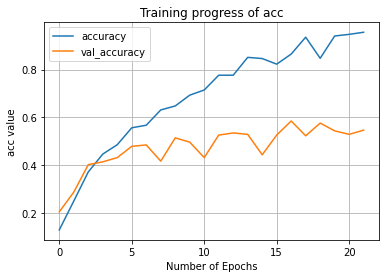
**Achint Chaudhary**

1. **CNN Architectures, Data Augmentation and Ensemble of Neural Networks**
2. **Experiments with variety of CNN Architectures**

**Note:** All our training settings in these parts use an SGD optimizer with an initial learning rate of 0.01. model.fit() is called with ReduceLROnPlateau() callback of Keras with patience of 5 epochs, this technique monitor a given metric will decrease the learning rate if metric under monitoring (validation loss in our case) does not improve and after some number of epochs passed by patience value. It allows us to set a moderate initial learning rate and changes it on the basis of the profile of validation loss. Also, EarlyStopping() Keras callback is used in experiments to prevent over-fitting by halting training when validation loss stops improving for some given number of epochs, in our case the number is set to 15.

* 1. **Baseline Architecture**

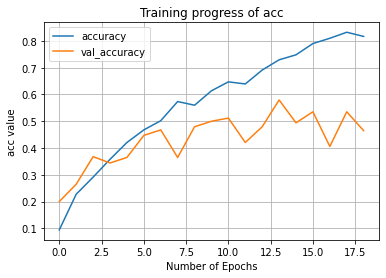
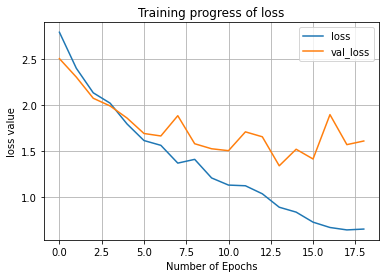
With suggested baseline model of one convolution Conv2D(32) and one hidden layer Dense(512), training terminates in around 25 epochs. 58.53% accuracy on the validation set is achieved using this simple model.

* 1. **An extra fully connected layer over baseline**

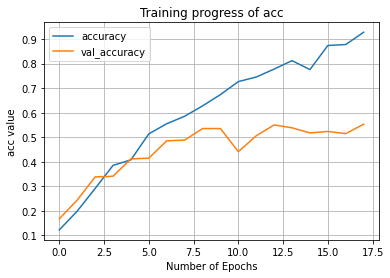
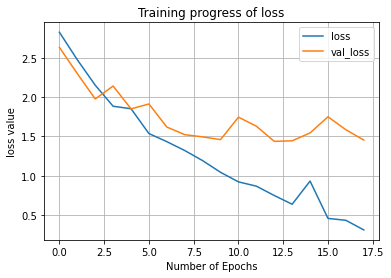
To increase the complexity of the classification system with a fully connected part from the baseline model, we add one more dense layer Dense(128). This actually reduces the performance of the network, as we have a very small dataset to train a large neural network onto.

Note: EarlyStopping is used to prevent over-fitting, but it does nothing but to halt training. Due to this reason validation accuracy is 57.94%.



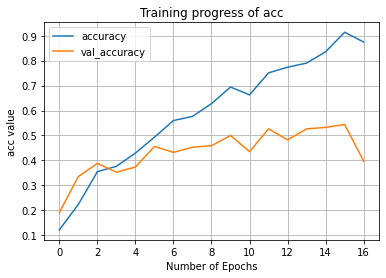
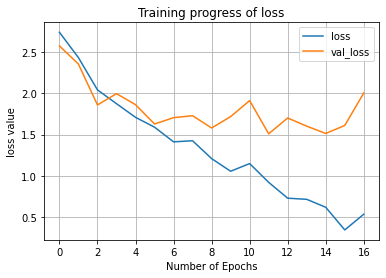
* 1. **Two convolution layers in baseline**

From the last experiment, we opt to keep only one hidden layer in fully connected part, but added one more convolution layer Conv2D(32). This experiment still resulted in a poor performance of 55.00% than the baseline performance of 58.53%.



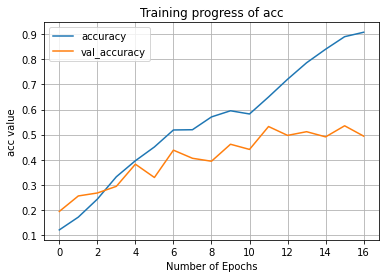
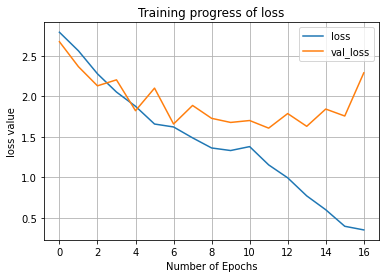
* 1. **Three convolution layers in baseline**

Extra convolution layers are expected to extract better spatial features, in a hope due to this reason, we further continue to create a model with three convolutions and one dense layer. This model obtains a performance of 53.64%.



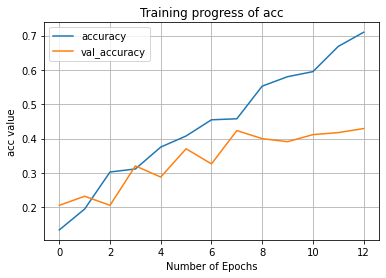
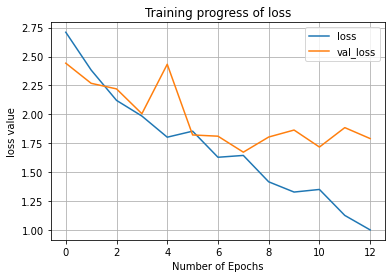
* 1. **Four convolution layers in baseline**

On one extreme, we experimented with a relatively much deeper network than the baseline with four convolution layers. This architecture inspired being much deeper, also shows less performance than previous all architectures, and obtained an accuracy of 53.24% only. Up to here we are not able to observe the popular statement of “Deeper is better” of deep learning.



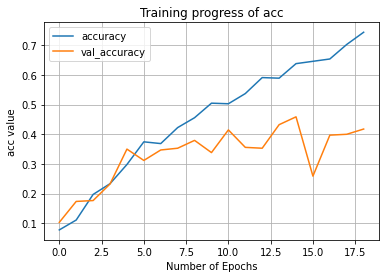
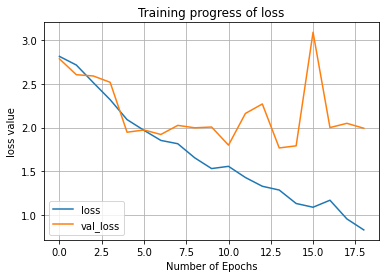
* 1. **Three convolution layers with two fully connected layers**

Now instead of just increasing the convolution layers, we chose architecture with more number of both convolution layer and fully connected layers over the baseline. But this even decreases performance on the validation set to an accuracy of 42.35%.



* 1. **VGG style four convolution layers with one fully connected layer**

Next we have experimented on a VGG style architecture inspired from literature and class lectures. With four convolution layers arranged in VGG style of 2 blocks (first block having two Conv2D(32), and second having two Conv2D(64)) and one dense layer same as baseline. Even architecture does not also outperformed baseline, and obtained an accuracy of 43.25% only.



Below Table 1., summarizes the validation accuracy for each model with baseline on top, and two deepest models with least performance on validation accuracy.

|  |  |  |
| --- | --- | --- |
| **Model Name** | **Architecture** | **Accuracy (in %)** |
| 0 | Conv2D [32,] + Dense [512,] | 58.53 |
| 1 | Conv2D [32,] + Dense [512,128,] | 57.94 |
| 2 | Conv2D [32,32] + Dense [512,] | 55.00 |
| 3 | Conv2D [32,32,32] + Dense [512,] | 52.64 |
| 4 | Conv2D [32,32,32,32] + Dense [512,] | 53.24 |
| 5 | Conv2D [32,32,32] + Dense [512,128] | 42.35 |
| 6 | Conv2D [32,32] + Conv2D [64,64] + Dense [512,] | 43.25 |

Table 1: Impact of Usage of Deeper architecture with smaller dataset

1. **Impact of Data Augmentation on Two Deepest networks**

All of the above experiments with CNN architectures are in increasing order of complexity of networks. We have observed that the deeper a model is, the poorer it performs on the validation set. This should not be the case with deep learning. The rationale behind this observation is the lack of a sufficient amount of training data.

In the case of computer vision tasks, Data augmentation is a popular technique to apply transformations (i.e., Rotation, Flipping, and Zooming) on dataset images in order to generate a dataset with more variation and is expected to increase the performance of models.

We have experimented with data augmentation on two of our deepest network (last two from Table 1). Resulted models trained using data augmentation outperformed all models trained without data augmentation.

|  |  |  |
| --- | --- | --- |
| **Model Name** | **Accuracy using Only Train data** | **Accuracy using Data Augmentation** |
| 5 | 42.35% | 65.59% |
| 6 | 43.25% | 61.18% |

Table 2: Impact of Augmentation on Two deepest models

Also we observed two common phenomena when models are trained using data augmentation as compared to when only original dataset images are used.

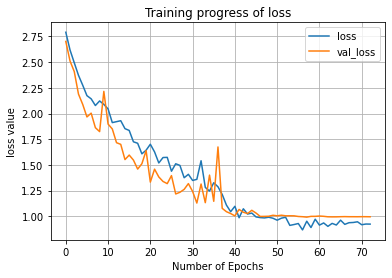
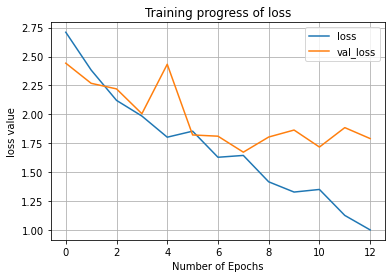
* 1. **Relatively better validation loss from training performance with augmentation**

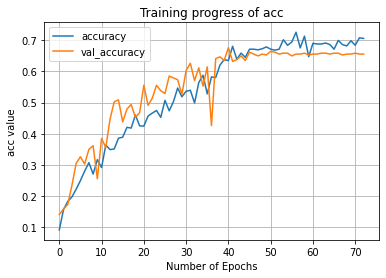
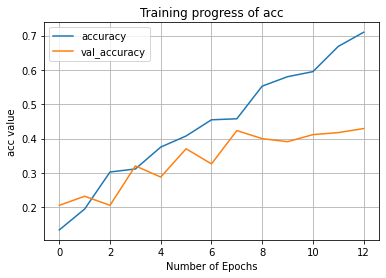
When training models from all of our design choices, validation loss generally is poor than training loss at the end of the training, the same trend is observed in terms of accuracy.

While, when training models with data augmentation, the scenario is the exact opposite, performance on the validation set is much better than that of the training dataset itself.

This implies that images in the validation set can be represented well with certain training images augmented properly with certain variations.

The below graph compares performance on loss function, as well as the accuracy of one of the deepest model when trained without and with data augmentation. Images with Data Augmentation are on the right hand side.





* 1. **Usage of ReduceLROnPlateau() callback**

Even deployed in training, this Keras callback is never seen to work when models are trained using a limited amount of original dataset.

But with data augmentation, the learning rate is decreased multiple times, and models hence also achieve fine-grained tuning, apart from dataset having more variations.

1. **Ensemble techniques of CNNs**

A very popular machine learning technique is to use an ensemble of different models to increase accuracy. The basis of this technique is the law of large numbers. Now we will discuss the three different model ensemble techniques explored.

* 1. **Simple Average**

Since we know that data augmentation is an important aspect to obtained higher validation accuracy, we have trained all of our seven design choices with data augmentation (keeping all other settings to be the same) before using the ensemble.

All CNN architectures have a Softmax layer at the end to decide which provided probability value for each class among the 17 flower classes.

In a simple ensemble method, the mean value of the probability vector of length 17 from each model in the ensemble can be taken. This simple technique provides us an accuracy of 68.82%, which is more than any of our seven baseline models. Hence, we can conclude that the simplest ensemble can also outperform all individual learners in it.

* 1. **Learned Weighted Average**

Inspired from the above method, we have looked into another possibility, where the output of all learners in the ensemble is not just simply being averaged. We have looked into a possibility where weights for each model are learned to decide the contribution of each model in ensemble towards the final output.

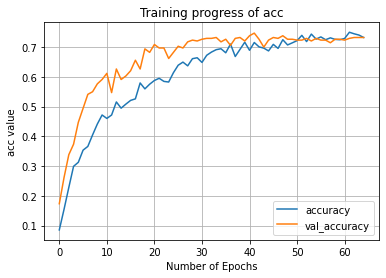
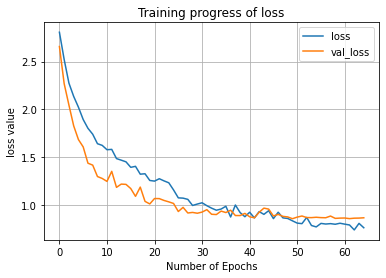
Weights for this learning scheme are learned from predictions on the training set to be maximized. While, applying weighted average ensemble with learned weights, we also achieve the same accuracy of 68.82% as before.

* 1. **Dropout**

Since in the above weight-based averaging scheme, we are only ensembling the final outputs of seven trained neural networks. Due to this reason when such an ensemble is deployed, complexity is generally 5x-7x times more than base learner with only a marginal improvement in validation accuracy of 68.82% from 67.64% which is of one of a base learner in weighted ensemble.

Another technique in literature to prevent over-fitting and is shown to behave like the ensemble of neurons at each layer within the neural network itself is called “Dropout”[1,2]. Dropout builds ensemble within a neural network from time of training itself, hence improves prediction quality significantly with less overhead increase in model complexity.

For our experiments we have created a model with around 2.5x time more complexity in terms of the number of parameters in both convolution and fully connected layer. This value is acceptable as compared to a 5x-7x times more complex ensemble. Also this less model which is less complex than average based ensemble achieves an accuracy of 72.65% on the validation set. This results in 5.01% times more accuracy with 2.5x times more complex network, instead of 1.18% times more accuracy using 7x times complex weight-based average ensemble. The below graph shows the training progress of the dropout based neural network.



The below table shows the performance of all individual learners with three ensemble techniques and their performance on validation accuracy.

|  |  |
| --- | --- |
| **Model** | **Validation Accuracy** |
| 0 | 65.29 |
| 1 | 61.76 |
| 2 | 67.64 |
| 3 | 64.70 |
| 4 | 67.64 |
| 5 | 66.47 |
| 6 | 67.35 |
| Simple Avg | 68.82 |
| Weighted Avg | 68.82 |
| Dropout Network | 72.65 |

Table 3: Comparison of ensemble techniques with individual learned models

1. **Feature Extraction, Transfer Learning with Fine Tuning**

One of the core limitations of our dataset is the insufficient amount of training data to train deep neural networks and achieve high validation accuracy. To overcome this limitation in this section we will be using pre-trained neural networks which are publicly available for non-commercial use and are trained on a large amount of training data with best practices of deep learning that win popular competitions like Image-Net.

1. **Feature extraction and usage of secondary ML algorithm**

In this part we have used the VGG-16 architecture to extract features from the last convolution layer of CNN. These features are flattened to create a vectorial representation for each image, and are provided to a variety of secondary machine learning techniques from Scikit-learn library.

Below table enlist the validation accuracy for a variety of secondary Machine learning algorithms. Best hyper-parameters of each of the following algorithms are obtained using Grid search on top of K-fold Cross-validation which evaluates model performance at each possible combination of hyper-parameters, also Stratified split is used to create validation splits during K-fold cross-validation to ensure same class ratio between any two splits.

|  |  |
| --- | --- |
| **Machine Learning Algorithm** | **Validation Accuracy (in %)** |
| Naïve Bayes | 62.94 |
| Linear Discriminant Analysis | 78.82 |
| K Nearest Neighbors | 79.41 |
| Logistic Regression | 89.41 |
| Support Vector Machine | 89.71 |
| Random Forest | 83.24 |
| Neural Network | 88.53 |
| Stacking Ensemble | 83.24 |

Table 4: Accuracy of various secondary Machine Learning Algorithms

For getting the best hyperparameter values for any of the algorithms, one can look at Jupyter notebook for this part of the assignment. It is notable that even simpler algorithms like Logistic regression are able to outperform with an accuracy of 89.41% over dropout based ensemble network trained on only the original dataset which has an accuracy of only 72.65%.

Also, we have tried a stacking based ensemble of all variety of secondary ML algorithms, each with their best set of hyper-parameters, but was not able to outperform many simpler ML algorithms itself like Logistic regression. Hence we can conclude that, with or without transfer learning coming into the picture. Simpler model ensemble techniques are not able to outperform significantly the individual learn in ensemble.

1. **Transfer Learning and Fine Tuning results**

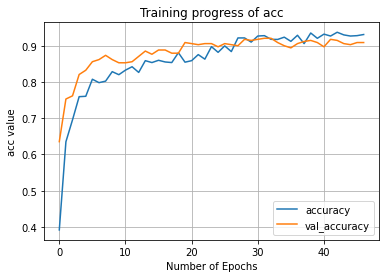
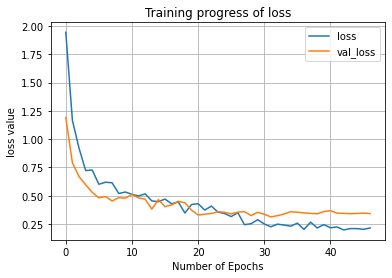
**Note:** All our training settings in these parts uses Adam optimizer (due to faster convergence, which is essential as limited timed sessions on Google Colab) with an initial learning rate of 0.001. Rest all experimentation settings are the same as Part A of the assignment.

In this experiment, we have attached a full convolution part on top of the full convolution base of VGG-19. The reason for switching from VGG-16 to VGG-19 is more scope of fine-tuning due to the extra number of convolution layers, which we will see in further subsections.

* 1. **Transfer Learning**

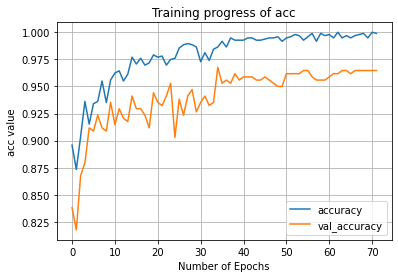
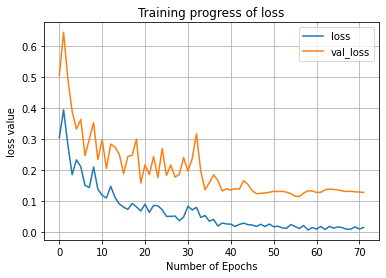
In this experiment we have connected a fully connected network on the convolution base of VGG-19. Convolution base is set to be non-trainable, and only provides extracted features to fully connected network on the top. Best hyper-parameters for the fully connected network are taken from an experiment done on neural networks as a secondary machine learning algorithm with Scikit-library in the previous section. Also a dropout is added to make this fully connected head network more powerful in terms of internal ensembling during training.

This overall experiment resulted in validation accuracy of 92.06% which is more than best obtained 89.71% using SVM in the previous experiment. Below is performance during training using transfer learning.



* 1. **Fine Tuning**

In first phase of fine tuning phase, last convolution block of VGG-19 is set to be trainable, with 10 time lesser learning rate than what is used in first experiment of transfer learning. This gains a more increase in validation accuracy up to 95.88%. Below graphs shows the training profile for fine-tuning the transfer learning based model.



In the next phase of fine-tuning one more convolution block from the last end of VGG-19 base is set to be trainable, and again a 10 times lesser learning rate from the previous experiment is used for fine-tuning. This model overall resulted in an accuracy of 96.47%.

Hence we can conclude that training a CNN from scratch is a bad practice, as it not only requires more resources to train and converge to meaningful values, but also provide far less performance than using transfer learning-based models when available pre-trained on a related task from what we have in hand.

|  |  |
| --- | --- |
| **Network Architecture** | **Validation Accuracy (in %)** |
| VGG19 (freezed) + Dense Hidden Layers(5) | 92.06 |
| VGG19 (Last 1 block active) + Dense Hidden Layers(5) | 95.88 |
| VGG19 (Last 1 block active) + Dense Hidden Layers(5) | 96.47 |

Table 5: Transfer learning and Fine tuning results using pre-trained VGG-19 CNN base

Note: Certain section for code common to part A and B of assignement are taken from [3].

1. **Introduction to Adversarial Machine Learning**
   1. **Introduction**

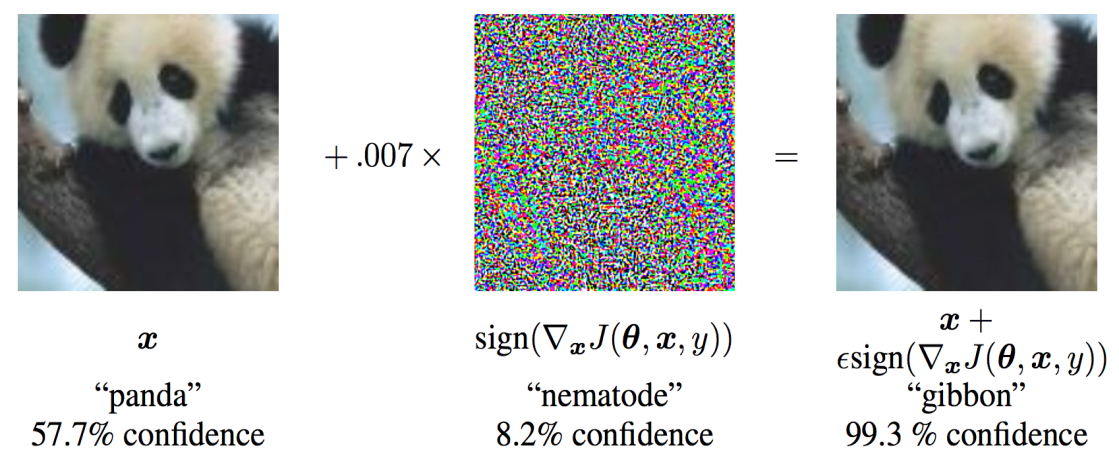
Machine and Deep learning applications are very common in recent times, and their applications will explore more into the future. With the advent of massively parallelization hardware like GPUs, scalable software frameworks like Tensor-flow, ever-growing high quality and large datasets coupled with better practices and continuous research, Machine, and Deep Learning applications have reached a superior level of performance for many tasks of which data comes from a natural source i.e., picture of animals.

However these algorithms are not robust to adversarial attacks, which make it difficult to provide guarantees on the performance behavior of a Machine learning system. Security depends on resources available to the adversary and their intention to disrupt the normal functioning of the system.

Security from adversary poses challenges to CIA properties of system (confidentiality, integrity, or availability). So, at par information from the system should not be leaked, the system should behave as expected and should continue to serve its purpose despite any kind of attack on it.

**Dataset Poisoning**

Adversary in such a case will provide images into a training dataset, which will tend to degrade prediction quality in real life [4]. As shown in [5], below is an image from a real dataset can be perturbated by adding very less carefully chosen noise, so as to change model prediction, even when adversarial example in indistinguishable from real image by a human.



This is achieved by pushing dataset images on the other side of the decision boundary approximated by the model during training. This kind of issue arises when an adversary can interact and query the oracle model repeatedly, hence can collect a wide variety of data and response of oracular model on each data. Later on training a model with the obtained dataset, craft adversarial examples using the model at hand, and put them to degrade the integrity of the oracular system.

* 1. **Robustness strategies**
     1. **Adversarial training**

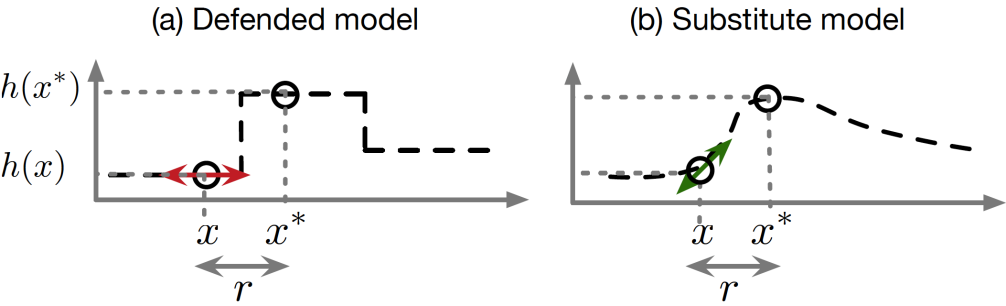
One approach can be to involve adversarial examples during training itself, this is a computationally expensive operation, and a simple fast gradient sign method (FSGM) for crafting simple adversarial examples is proposed by [4]. This increases robustness against simple adversarial attacks.

* + 1. **Defensive distillation**

The decision boundary of the model should be smoother to prevent adversarial attacks, and we can achieve the same by training a model on soft labels produced by another training model before that on discrete output labels during classification [7]. Model trained this way on soft labels is tends to be more robust against attacks like FSGM and Jacobian-base saliency map approach.

* + 1. **Gradient Masking**

Gradient and probability values for classification among different classes provide adversary some information about the model and help him to craft adversarial examples. An approach suggested in [8] is to only notify adversary about most likely class, instead of probability values, which sounds to make differential attacks impossible due to lack of gradient.



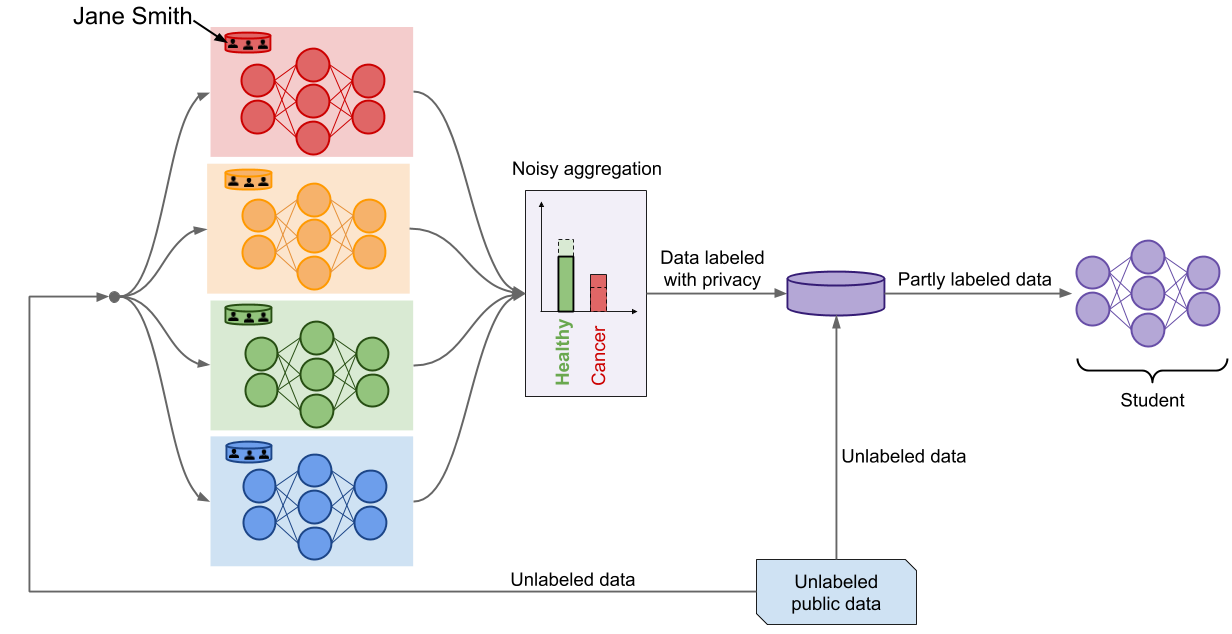
However, this strategy failed as the adversary can get outputs from the oracular system on a large set of images, and then train a copy of model at her end, and use that to craft adversarial examples as shown in figure above.

* 1. **Privacy preserving Machine Learning**

While dataset poisoning is intended to break the machine learning system by sending data crafted in an adversarial manner. Breaking into privacy of machine learning systems to obtain information about statistical properties of the dataset can leak data regarding biases into the dataset used to train the system, or may leak sensitive information also. This can be obtained by crafting examples that closely resemble data points used to training the oracular ML system [6].

Private Aggregation of Teacher Ensembles (PATE) is a technique proposed by [9]

to provide privacy in machine learning frameworks. It works on a simple intuition that if multiple models are trained on different partitions of training dataset then a decision made as a voting classifier among all these models cannot be utilized to extract any information about specific data point which will lie in either of the models voting for the output class. Extending to the same idea and provide cost-efficient design, a student network is learned on soft labels of noisy aggregation obtained from all of the teacher models trained on different partitions on training dataset by repeatedly querying teacher ensemble. This technique is diagrammatically shown below with a specific example of a movie named “Jane Smith” on the IMDb dataset available over the internet.



Gradient values of the Student model after training cannot be utilized by an adversary to obtaining membership check for any data point into a training dataset. Working with this multi-teacher and single-student framework provides the same efficiency as of using a single model, and also preserves the privacy of the machine learning dataset on which the entire model is trained on.

* 1. **Conclusion**

In the end we can conclude that it is possible to provide privacy into Machine learning systems, but absolute guarantee against attack cannot be provided and always depend upon computational capability of adversary.

**References**

1. Dropout: a simple way to prevent neural networks from overfitting, Hinton et al., The Journal of Machine Learning Research (ACM), 2014
2. Analysis of Dropout Learning Regarded as Ensemble Learning, Hara et al., International Conference on Artificial Neural Networks, 2016
3. Code for data augmentation taken from <https://keras.io/examples/cifar10_cnn/>
4. Intriguing properties of neural networks, Goodfellow et al., 2013
5. Explaining and harnessing adversarial examples, Goodfellow et al., 2014
6. Towards the Science of Security and Privacy in Machine Learning, Papernot et al., 2016
7. Distilling the knowledge in a neural network, Hinton et al., 2015
8. Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples”, Papernot et al., 2016
9. Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data, Papernot et al., 2017
10. Image courtesy for Adversarial ML report to <https://cleverhans.readthedocs.io/>