**Keras Installation**

I used Keras for building deep neural network. It is a high-level library that operates on top of the TensorFlow. The perquisites for getting started with Keras is Python. I recommend installing Anaconda , a Python distribution. It is compatible with Python 2.7 – 3.5. I used Jupyter notebook for Python. By default, Keras uses TensorFlow as a backend.

I followed following steps for the installation: -

1. Download Anaconda for your platform: [https://www.anaconda.com/download](https://www.anaconda.com/download/#macos)
2. Install TensorFlow i.e. pip install TensorFlow
3. Install Keras i.e. pip install Keras

**System Overview**

The image dataset consists of good, blurry and flare image types with each type containing 25 images. I splitted the dataset into training and validation sets, where the training data has 20 images and the validation data has 5 images of each type.

Given the nature of the dataset, CNN was chosen since it’s the default neural network choice for working with images. However, given the size of data set, I used concept of one-shot learning i.e. Siamese Networks with CNN Base network. The Siamese network accepts the pair of images as an input where the images are encoded into feature vectors and the Euclidean distance between the feature vectors is used to predict the label for a given image. We can solve the problem of paired images by using the binary classification approach where the probability of same images is 1 and probability of different images is 0. The classification problem takes the element wise difference of the feature vector and applies binary classification to the difference.

I used validation data to compute the performance of model. Accuracy was chosen as a performance metric for this classification problem. The given image from the validation data was compared with the random list of training images. The image was assigned a label of the training image for which the Euclidean distance between the feature vectors of the paired images is minimum.

**Outline of Algorithm**

* The Siamese network accepts the **paired input**. The paired input contains the list of positive pairs i.e. same kind of images and the negative pairs i.e. different images. I created the paired inputs and assigned the target value of 1 for the positive pairs and 0 for the negative pairs.
* I used Siamese Network with CNN as the base network. The CNN has **3 convolutional layers** with the size of 64,128,256 neurons respectively.
* **Batch normalization** is used for data pre-processing i.e. before feeding the data into an input layer. It normalizes and standardizes the data to the same scale. It enhances the training speed and helps to reduce the problem of exploding gradients.
* I used **Relu** as the activation function that transforms the input to max(0,x). It means the more positive the neuron the more activated it is.
* The **dropout regularization** technique is used for reducing the overfitting in neural networks.
* The model does not include any **padding**. Therefore, the original dimensions of image are reduced through MaxPooling and filters. Thus, reducing the number of trainable parameters.
* The total number of **trainable parameters** for my model are 44M and **total memory** per image is nearly 30Mb for forward network.
* **Weight and Bias Initialization** - The random samples are drawn from a normal Gaussian distribution for the initialization.
* **Optimiser -** I used Adadelta optimizer with the learning rate of 0.0001.
* **Contrastive Loss** - The contrastive loss function used in Siamese network tends to increase distance between the negative pairs and decreases the distance between the positive pairs.

**Loss function**

When y\_true is 1, that means the images are duplicates of each other, so the Euclidean distance (y\_pred) between their outputs must be minimized. So the loss is taken as the square of that Euclidean distance itself - K.square(y\_pred).

When y\_true is 0, i.e. the images are not duplicates, then the Euclidean distance between them must be maximized, at least to the margin. So, the loss to be minimized is the difference of the margin and the Euclidean distance - (margin - y\_pred). If the Euclidean distance (y\_pred) is already greater than the margin, then nothing is to be learned, so the loss is made to be zero in that case by saying K.maximum(margin - y\_pred, 0).The margin parameter will reduce the difference between same images and increase the difference between different images. Thus, margin can be defined as the hyperparameter that is tuned when the model is trained.

**Intuition behind algorithm**

The most popular algorithm of Image classification is CNN. However, the major challenge with CNNs is the large amounts of data required to work properly. The given training data is very small. Therefore, I used a variant as Siamese Networks known to work well for small amounts of data.

**Summary of results**

Since the task involved is a classification one, I have used accuracy as the performance metric for the task. Since One-Shot learning focuses on the ability to classify a large number of classes based on lesser amounts of data, the accuracy is tested for different number of classes.

I used the Validation Data to compute the accuracy of model. The Validation Accuracy of Siamese model is 78 percent. The model is able to classify the good images, blurry images and flare images with overall accuracy of 78 percent.

**Potential future work**

The early layers in the network detect more generalized patterns like edges, shapes. The deeper we go in the network; the filters get more sophisticated and can detect more complex patterns like texture. So, adding more layers will improve the detection of blur and flare in the images.

I created a simple CNN due to the restricted computational resources. Given the good computational resources, the model’s accuracy can be improved by adding more layers and increasing the number of filters.

Secondly, I created CNN with plain layers. Given more time, I would have tried adding Residual network layers instead of plain layers in CNN. The layers added in ResNet way helps to improve the performance by solving the problem of exploding/vanishing gradients.