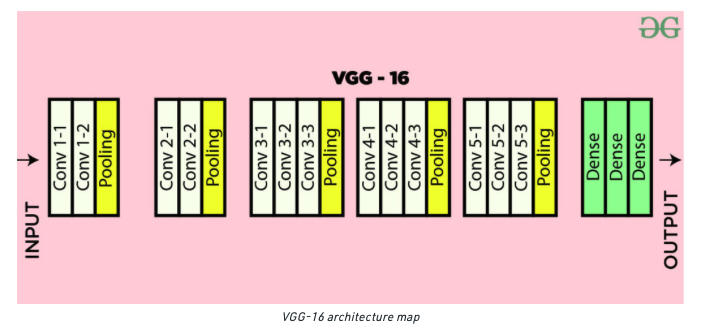
**Face Recognition based on Fine Tuned VGG 16 Network**

**Train Test Split and Data Augmentation:**

1. Training: Randomly select any 4 poses out of 9 poses for each subject.
2. Perform data augmentation on the randomly selected 4 poses to generate 99 augmented images per pose. Thus, there will be 400 images per subject for training the network. Total 400 \* 200 = 80000 images in the training set.
3. The remaining 5 poses per subject will be used for testing. There is no need to perform data augmentation on the test set. Thus, there will be a total of (5\*200 = 1000) images in the test set.

You may consider using Keras Image Data Generator API for data augmentation.



**Network Architecture:**

1. Use the first 15 layers of the pretrained VGG-16 architecture shown above. This means that weights of the first 15 layers of the pretrained VGG-16 architecture are freezed.

[Note that pooling is not counted as a layer.]

1. After the 15th layer, add the below fully connected layers for fine tuning the network
   1. 16th Layer – Fully Connected Layer with 2048 Neurons [Activation Function: ReLu, Dropout Rate: 0.25]
   2. 17th Layer – Fully Connected Layer with 1024 Neurons [Activation Function: ReLu, Dropout Rate: 0.25]
   3. 18th Layer – Fully Connected Layer with 512 Neurons [Activation Function: ReLu, Dropout Rate: 0.25]
   4. 19th Layer – Fully Connected Layer with 200 Neurons [If 200 subjects are being considered] [Activation Function: Softmax, Dropout Rate: 0.25]
2. Loss Function to be used: Categorical\_Crossentropy [Used for multi class classification]
3. Train the above network architecture. The input to the network is a face image of size 224×224 with the average face image (computed from the training set) subtracted.
4. Include Early Stopping in the Code with patience level of 20 Epochs, Code Ref: <https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping>
   1. Plot Accuracy vs Epochs curve for train and test dataset
   2. Plot Loss vs Epochs curve for train and test datasets

**When preparing an image array for face recognition using a pretrained/fine tuned VGG16 model, you can follow these preprocessing steps:**

1. Resize the image: VGG16 models typically require input images of a fixed size. Resize your image to the desired input size of the VGG16 model. For example, VGG16 often takes input images of size 224x224 pixels. You can use libraries like OpenCV or Pillow to resize the image.
2. Convert to RGB: Ensure that the image is in RGB format. If the image is in a different color format, such as BGR, convert it to RGB. This can be done using the appropriate functions provided by the image processing libraries.
3. Normalize the pixel values: Normalize the pixel values of the image to a range of 0 to 1. Divide the pixel values by 255 to achieve this normalization.
4. Standardization: Subtract the mean pixel values from the image. The mean values can be calculated based on the dataset that the VGG16 model was trained on. For face recognition, you can use the mean values obtained from the face dataset used during the training of the VGG16 model.

When fine-tuning the VGG16 model on a different dataset, it is recommended to calculate the mean values specific to your fine-tuning dataset. This can be done by computing the mean pixel values across the images in your dataset.

Once you have the mean values for each RGB channel, you can subtract them from the corresponding channels of your input images during the preprocessing step. This helps to center the data and remove any bias due to different lighting conditions.

1. Expand dimensions: Expand the dimensions of the image array to match the expected input shape of the VGG16 model. VGG16 models typically expect a 4D input of shape (batch\_size, height, width, channels). You can use libraries like NumPy to expand the dimensions of the image array.
2. Data type conversion: Convert the image array to the appropriate data type expected by the model, such as float32.

Once you have preprocessed the image array according to these steps, you can pass it as input to the pretrained VGG16 model for face recognition.

Note: There are several implementations of pretrained / fine tuned VGG 16 model on the internet. Please feel free to refer them including the pre-processing steps.

**Biometric System Performance Evaluation:**

1. Calculate the Total Genuine Attempts and Genuine Pairs assuming that there are only 5 images per subject [Test Images]
2. Calculate the Total Impostor Attempts and Impostor Pairs assuming that there are only 5 images per subject [Test Images]
3. Compute GAR, FAR, FRR, EER. Plot DET Curve, ROC Curve, Genuine and Impostor Scores Distribution Curve.

**Note on Early Stopping:**

Ref: <https://machinelearningmastery.com/early-stopping-to-avoid-overtraining-neural-network-models/>

A major challenge in training neural networks is how long to train them.

Too little training will mean that the model will underfit the train and the test sets. Too much training will mean that the model will overfit the training dataset and have poor performance on the test set.

A compromise is to train on the training dataset but to stop training at the point when performance on a validation dataset starts to degrade. This simple, effective, and widely used approach to training neural networks is called early stopping.

It is also common to use the loss on a validation dataset as the metric to monitor, although you may also use prediction error in the case of regression, or accuracy in the case of classification.