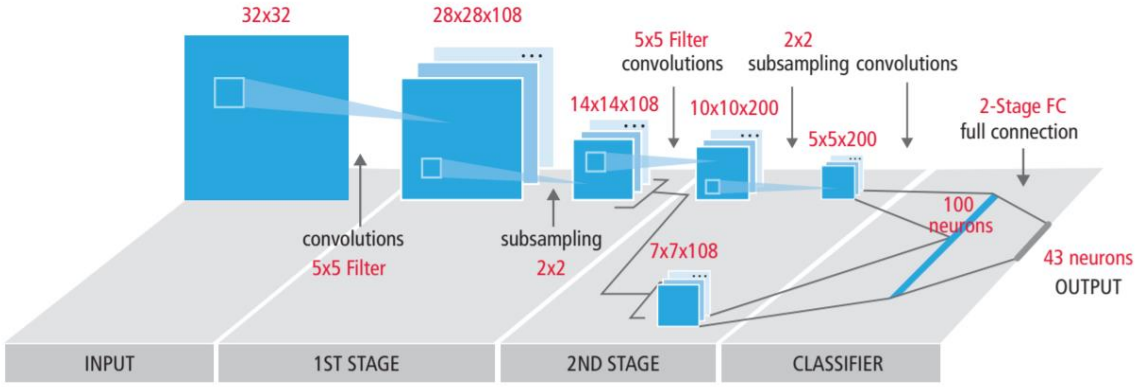
MAJOR PROJECT DOCUMENTATION

1) Which Neural Network and why?

A neural network is a system of interconnected artificial "neurons" that exchange messages between each other. The connections have numeric weights that are tuned during the training process, so that a trained network will respond correctly when presented with an image or pattern to recognize. The network consists of multiple layers of feature-detecting "neurons". Each layer has many neurons that respond to different combinations of inputs from the previous layers.

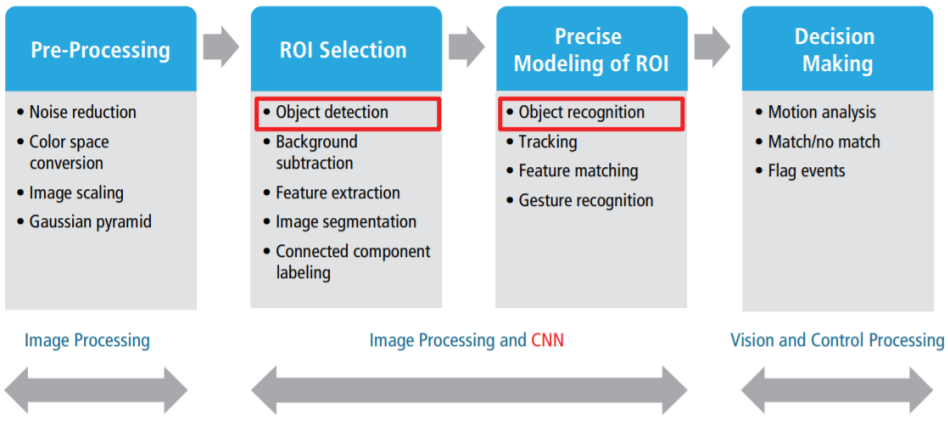
Given the project is emotion recognition, the neural network used here is Convolutional Neural Network. Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven highly effective in areas such as image recognition and classification. In a CNN, convolution layers play the role of feature extractor. But they are not hand designed. Convolution filter kernel weights are decided on as part of the training process. Convolutional layers can extract the local features because they restrict the receptive fields of the hidden layers to be local.

Typical example:



Why CNN?

- Ruggedness to shifts and distortion in the image
- Fewer memory requirements
- Easier and better training



I have made a 7 blocks neural network with hidden units in each layer such that: (BLK1: 32 , BLK2: 64 , BLK3: 128 , BLK4: 256 , BLK5,6,7: Fully connected classifier) and used Adam optimiser. The **Sequential model** is employed to easily stack sequential layers (and even recurrent layers) of the network in order from input to output. I have used 25 epochs and the data is split into training data and validation data inside the dataset itself. Batch size of 32 has been selected in the model. **Elu & softma**x activation has been used in the hidden units.

2) Which optimizer and why?

Neural networks train for many rounds known as epochs. In each batch of these epochs, the neural network readjusts its learnt weights to weights to reduce its loss from the previous epoch. The process of learning these weights is achieved by an optimizer.it helps to minimize the loss function (difference between model output and target output). Here I have used ADAM optimizer. (Adaptive Moment Estimation) Adam is an adaptive learning rate method, which means, it computes individual learning rates for different parameters. It uses estimations of first and second moments of gradient* to adapt the learning rate for each weight of the neural network. (*the gradient is a numeric calculation allowing us to know how to adjust the parameters of a network in such a way that its output deviation is minimized.) It functions well with sparse datasets.

3) Which accuracy metric and why?

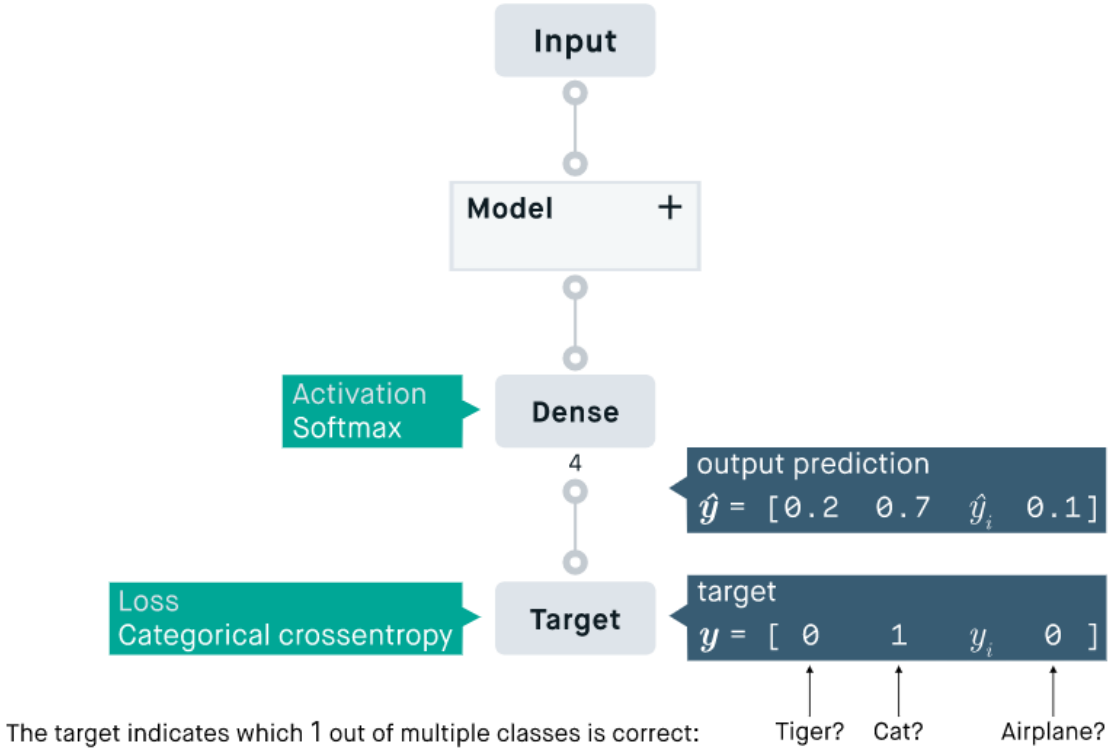
The accuracy metric used here is **accuracy**. Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition: "In multilabel classification, this function computes subset accuracy: the set of labels predicted for a sample must *exactly* match the corresponding set of labels in y_true."

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

4) Which loss function and why?

Loss function is used as measurement of how good a prediction model does in terms of being able to predict the expected outcome. Also called **Softmax Loss**. It is a **Softmax activation** plus a **Cross-Entropy loss**. Categorical cross-entropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.

Formally, it is designed to quantify the difference between two probability distributions. It is commonly used metric in classification tasks, especially in computer vision with convolution neural networks.



5) Brief information on how cleaning was done. (if any)

No, the dataset was used as it is.

6) How data was got into the right shape? (if any)

To make the most of our few training examples, we will "augment" them via several random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better. The images were resized using ImageDataGenerator, which is used for real-time data augmentation.

- rotation_range is a value in degrees (0-180), a range within which to randomly rotate pictures
- width_shift and height_shift are ranges (as a fraction of total width or height) within which to randomly translate pictures vertically or horizontally
- rescale is a value by which we will multiply the data before any other processing. Our original images consist in RGB coefficients in the 0-255, but such values would be too high for our models to process (given a typical learning rate), so we target values between 0 and 1 instead by scaling with a 1/255. factor.
- shear_range is for randomly applying shearing transformations
- zoom_range is for randomly zooming inside pictures
- horizontal_flip is for randomly flipping half of the images horizontally --relevant when there are no assumptions of horizontal asymmetry (e.g. real-world pictures).
- fill_mode is the strategy used for filling in newly created pixels, which can appear after a rotation or a width/height shift.

7) What functions/features of OpenCV were used?

To make use of computer vision in the project we used OpenCV. Various functions of OpenCV were used and here are few of them:

- cv2.VideoCapture(0) - To capture a video, you need to create a VideoCapture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. So, simply pass 0 (or -1). You can select the second camera by passing 1 and so on.
- cv2.cvtColor(input_image, flag) - For BGR → Gray conversion we use the flags cv2.COLOR_BGR2GRAY. Similarly, for BGR → HSV, we use the flag cv2.COLOR_BGR2HSV. Flag parameter determines what type of conversion will take place.
- cv2.rectangle(img, pt1, pt2, colour, thickness) - The function draws a rectangle outline or a filled rectangle whose two opposite corners are pt1 and pt2.
- cv.resize(src, dsize, interpolation) - The function resize resizes the image src down to or up to the specified size. Note that the initial dst type or size are not considered. Instead, the size and type are derived from the src, dsize, fx, and fy.
- cv.putText(img, text, org, fontFace, fontScale, color, thickness) - The function cv.putText renders the specified text string in the image. Symbols that cannot be rendered using the specified font are replaced by question marks.
- cv2.imshow() – We use the function cv2.imshow() to display an image in a window. The window automatically fits to the image size. First argument is a window name which is a string. Second argument is our image. You can create as many windows as you wish, but with different window names.
- cv2.waitKey() - cv2.waitKey() is a keyboard binding function. Its argument is the time in milliseconds. The function waits for specified milliseconds for any keyboard event. If you press any key in that time, the program continues. If 0 is passed, it waits indefinitely for a key stroke. It can also be set to detect specific keystrokes.
- cv2.destroyAllWindows() - cv2.destroyAllWindows() simply destroys all the windows we created. If you want to destroy any specific window, use the function cv2.destroyWindow() where you pass the exact window name as the argument.

8) Which dataset have you used?

https://www.kaggle.com/deadskull7/fer2013

https://datarepository.wolframcloud.com/resources/FER-2013