**1. Convolutional Neural Networks (CNNs)**

CNNs will be used for feature extraction from images or video frames. They’re goodl at capturing spatial hierarchies in data and identifying body parts and their positions in yoga poses.

**Components:**

- **Input Layer:** Accepts images or frames from a video.

- **Convolutional Layers:** Apply filters to the input image to detect various features (edges, textures, shapes). These layers help recognize different parts of the body.

- **Pooling Layers:** Reduce the spatial dimensions of the feature maps, retaining essential information and reducing computational load.

- **Input Nodes:** After several convolutional and pooling layers, fully connected layers aggregate the features learned and predict the yoga poses.

- **Output Layer:** Outputs key points or pose classes.

**2. Recurrent Neural Networks (RNNs)**

RNNs, specifically Long Short-Term Memory (LSTM) networks, can handle sequential data like video frames. They can learn temporal dependencies and provide insights into the flow and transition of poses.

**Components:**

- **Input Layer:** Accepts sequences of feature maps generated by the CNN.

- **LSTM Layers:** Process the sequential data to capture temporal patterns and dependencies in the movements.

- **Fully Connected Layers:** After the LSTM layers, fully connected layers interpret the temporal patterns and make predictions about the sequence of poses.

- **Output Layer:** Outputs the recognized sequence of yoga poses.

**3.**  **Human-Joint Localization (Pose Estimation Model)**

The primary role of this model is to detect and localize the key joints of the human body (e.g., shoulders, elbows, knees, etc.) in each image or video frame, identifying posture flaws.

**Components:**

- **Input Layer:** Accepts images or frames from a video.

- **Convolutional Layers:** Extract features from the input image.

- **Heatmap Generation Layers:** Generate heatmaps for each key joint, indicating the probability of each joint’s location.

- **Post-Processing:** Extract the coordinates of the key joints from the heatmaps.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Implementation Workflow:**

1. **Data Collection and Preprocessing:** Gather a dataset of images and videos of yoga poses. Annotate key points or pose labels. Preprocess the data (resize images, normalize pixel values).

2. **CNN Feature Extraction:** Pass each image or frame to extract spatial features through the CNN.

3. **Temporal Modeling with RNN:** Feed the sequence of feature maps into the RNN to capture temporal dependencies and recognize the sequence of poses.

4. **Training:** Train the combined CNN-RNN model using a labelled dataset. Use loss functions suitable for pose estimation (e.g., mean squared error for key points) and pose classification (e.g., cross-entropy loss).

5. **Evaluation and Tuning:** Evaluate the model’s performance on a validation set. Fine-tune hyperparameters and model architecture based on performance metrics (accuracy, precision, recall).

**Potential Issues**

- **Variability in Poses:** Ensure the dataset is diverse enough to capture different pose variations.

- **Occlusion:** Address cases where parts of the body are hidden or overlap.

- **Computational Complexity:** Optimize the model to balance accuracy and computational efficiency.

**ML techniques:** Xception, VGGNet, & SqueezNet

**CNN + ML techniques (**with **LDA and GDA) = ML models**

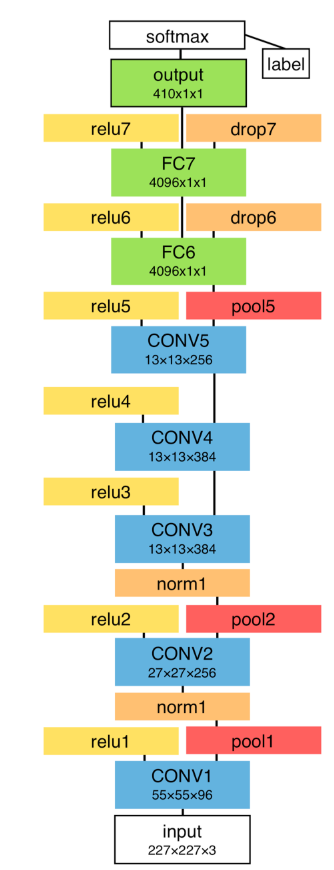
**ML models: LGDeep (most efficient), GDeep, LDeep, & Deep.**

For our yoga pose classification project, we utilize **Convolutional Neural Networks** (CNNs) for feature extraction from images. This is a multiclass classification problem since the input yoga pose will be classified into one of 107 yoga poses. Our CNN model will receive input of a square image, and output a vector of probabilities of each yoga pose match. As an activity classification problem, we are inspired by the research work of Amy Bearman et al. Figure below is the architecture used in Bearman’s research, and we aim to use it as a starting point. CNNs are suitable for this yoga classification project due to heir ability to automatically learn relevant features from raw image data, reducing the need for manual feature engineering. Their convolutional layers can detect and recognize patterns such as edges, textures, and shapes that are critical for distinguishing between different yoga poses, ensuring high accuracy.

A rough architecture of CNN should essentially have 5 main layers:

* Input
* Convolution (Filters)
* Pooling
* Input Nodes
* Output

The architecture begins with an **input layer** that accepts images. Following the input layer, several **convolutional layers** apply filters to the images, detecting various features such as edges, textures, and shapes. These layers are vital to recognizing different parts of the body. **Pooling** layers are then used to reduce the spatial dimensions of the feature maps, converting into the **input nodes** while reducing the computational load. After the convolutional and pooling layers, those input nodes aggregate the features learned and make predictions about the yoga poses. The **output layer** provides the final predictions, either in the form of key points (speicific joint locations) or pose classes (the type of yoga pose).



For predicting the sequential data from video frames, we utilize **Recurrent Neural Networks (RNNs),** specifically Long Short-Term Memory (LSTM) networks. LSTMs are necessary for learning temporal dependencies, which allows them to provide insights into the flow and transition of poses over time. The architecture begins with an input layer that accepts sequences of feature maps generated by the CNN (tentatively speaking). These feature maps contain spatial information extracted from the individual frames. The LSTM layers, composed of three main gates - forget gate, input gate, and output gate - process the sequential data [(Graves, 2013)](https://www.cs.toronto.edu/~graves/preprint.pdf). The forget gate decides which information to discard from the previous state, the input gate updates the cell state with new information, and the output gate determines the output based on the cell state. After the LSTM layers, the output layer provides the final recognized sequence of yoga poses.