### **1. Layers and Modules**

The core data structures of torch.nn are Module and its subclasses. A Module can represent a layer in a neural network, a complete network itself, or any component that has learnable parameters. Here are some examples of layers and network components you can find in torch.nn:

* **Linear layers** (nn.Linear): Fully connected (dense) layers.
* **Convolution Layers** (nn.Conv2d, nn.Conv3d, etc.): Convolutional layers used primarily in image processing.
* **Recurrent Layers** (nn.LSTM, nn.GRU): For sequence processing like time series analysis or natural language processing.
* **Transformer Layers** (nn.Transformer): Building blocks for transformer models.

### **2. Activation Functions**

Activation functions are crucial for introducing non-linearities into the network, which allows for learning more complex patterns. Common activation functions include:

* **ReLU** (nn.ReLU): A widely used, simple non-linearity.
* **Sigmoid** (nn.Sigmoid)
* **Tanh** (nn.Tanh)

### **3. Loss Functions**

To train a network, you need a way to measure how well the network is performing. torch.nn provides several loss functions to cater to different types of tasks like classification, regression, etc. Some examples are:

* **Mean Squared Error** (nn.MSELoss): Commonly used for regression tasks.
* **Cross Entropy Loss** (nn.CrossEntropyLoss): Widely used for classification tasks.

### **4. Utilities**

Besides layers and functions, torch.nn also includes utilities to help manage and manipulate these layers and models, such as:

* **nn.Sequential**: A simple way to build networks where a layer feeds directly into the next.
* **nn.ModuleList** and **nn.ModuleDict**: These are specially optimized lists and dictionaries that can contain modules and manage their parameters appropriately.

### **5. Normalization Layers**

Normalization layers help in stabilizing the learning process and include:

* **Batch Normalization** (nn.BatchNorm2d): Normalizes the input of each batch.
* **Layer Normalization** (nn.LayerNorm): Normalizes the input across the features.