Explanation of the below CNN code block:

```
n_classes = 24
kernel_size = 3
flattened_img_size = 75 * 3 * 3
model = nn.Sequential(
    # First convolution
    nn.Conv2d(IMG_CHS, 25, kernel_size, stride=1, padding=1),  # 25 x 28 x 28
    nn.BatchNorm2d(25),
    nn.ReLU(),
    nn.MaxPool2d(2, stride=2),  # 25 x 14 x 14
    # Second convolution
    nn.Conv2d(25, 50, kernel_size, stride=1, padding=1),  # 50 x 14 x 14
    nn.BatchNorm2d(50),
    nn.ReLU(),
    nn.Dropout(.2),
    nn.MaxPool2d(2, stride=2),  # 50 x 7 x 7
    # Third convolution
    nn.Conv2d(50, 75, kernel_size, stride=1, padding=1),  # 75 x 7 x 7
    nn.BatchNorm2d(75),
    nn.ReLU(),
    nn.MaxPool2d(2, stride=2),  # 75 x 3 x 3
    # Flatten to Dense
    nn.Flatten(),
    nn.Linear(flattened_img_size, 512),
    nn.Dropout(.3),
    nn.ReLU(),
    nn.Linear(512, n_classes)
)
```

For each convolution and pooling layer, the output shape depends on the input shape, kernel size, stride, and padding.

1. Convolution Layer Output Shape

The output shape for a 2D convolutional layer can be calculated as follows for each dimension (height and width):

$$Output\ Dimension = \frac{\text{Input Dimension} - \text{Kernel Size} + 2 \times \text{Padding}}{\text{Stride}} + 1$$

where:

- **Input Dimension** is the height or width of the input.
- **Kernel Size** is the size of the convolutional kernel (filter).
- **Padding** is the number of pixels added around the input.
- **Stride** is the step size of the kernel.

The number of output channels (depth) is determined by the number of filters applied, not affected by the height and width calculation.

2. Max Pooling Layer Output Shape

For max pooling layers, the output shape can be calculated similarly:

$$\text{Output Dimension} = \frac{\text{Input Dimension} - \text{Pool Size}}{\text{Stride}} + 1$$

where **Pool Size** is the size of the pooling window.

Walkthrough of Each Layer

Let's go layer by layer using these formulas.

Layer 1: First Convolution Block

1. **Input Shape**: `IMG_CHS x 28 x 28`
2. **Conv2d Layer**: `nn.Conv2d(IMG_CHS, 25, kernel_size=3, stride=1, padding=1)`
   - Kernel Size = 3
   - Padding = 1
   - Stride = 1
   - **Output Shape Calculation**:
     - Height and Width: $\frac{28-3+2\times1}{1} + 1 = 28$
     - The output shape is `25 x 28 x 28` (25 is the number of output channels).
3. **MaxPool2d Layer**: `nn.MaxPool2d(2, stride=2)`
   - Pool Size = 2
   - Stride = 2
   - **Output Shape Calculation**:
     - Height and Width: $\frac{28-2}{2} + 1 = 14$
     - The output shape is `25 x 14 x 14`.

Layer 2: Second Convolution Block

1. **Input Shape**: `25 x 14 x 14`
2. **Conv2d Layer**: `nn.Conv2d(25, 50, kernel_size=3, stride=1, padding=1)`
   - Kernel Size = 3
   - Padding = 1
   - Stride = 1

- **Output Shape Calculation**:
  - Height and Width: $\frac{14-3+2\times1}{1}+1=14$
  - The output shape is `50 x 14 x 14`.

3. **MaxPool2d Layer**: `nn.MaxPool2d(2, stride=2)`
   - Pool Size = 2
   - Stride = 2
   - **Output Shape Calculation**:
     - Height and Width: $\frac{14-2}{2}+1=7$
     - The output shape is `50 x 7 x 7`.

Layer 3: Third Convolution Block

1. **Input Shape**: `50 x 7 x 7`
2. **Conv2d Layer**: `nn.Conv2d(50, 75, kernel_size=3, stride=1, padding=1)`
   - Kernel Size = 3
   - Padding = 1
   - Stride = 1
   - **Output Shape Calculation**:
     - Height and Width: $\frac{7-3+2\times1}{1}+1=7$
     - The output shape is `75 x 7 x 7`.
3. **MaxPool2d Layer**: `nn.MaxPool2d(2, stride=2)`
   - Pool Size = 2
   - Stride = 2
   - **Output Shape Calculation**:
     - Height and Width: $\frac{7-2}{2}+1=3$
     - The output shape is `75 x 3 x 3`.

Flatten Layer

1. **Input Shape**: `75 x 3 x 3`
2. **Flattening**: Converts this 3D tensor to a 1D tensor with size `75 * 3 * 3 = 675`.

Fully Connected (Dense) Layers

1. **Linear Layer**: `nn.Linear(675, 512)`
   - Input: 675 (flattened image size)

    - Output: 512 (size of the hidden layer)
2. **Final Linear Layer**: `nn.Linear(512, n_classes)`
    - Input: 512
    - Output: `n_classes (24)`