The code downloads a pre-trained VGG16 model and extract the input, keep probability, layer 3, layer 4 and layer 7 from it (method `load_vgg` from line 20 to line 44). Those layers are used in the `layers` to create the rest of the network:

- One convolutional layer with kernel 1 from VGG's layer 7 (line 62).
- One deconvolutional layer with kernel 4 and stride 2 from the first convolutional layer (line 70).
- One convolutional layer with kernel 1 from VGG's layer 4 (line 78).
- The two layers above are added to create the first skip layer (line 86).
- One deconvolutional layer with kernel 4 and stride 2 from the first ship layer (line 88).
- One convolutional layer with kernel 1 from VGG's layer 3 (line 96).
- The two layers above are added to create the second skip layer (line 104).
- One deconvolutional layer with kernel 16 and stride 8 from the second skip layer (line 106).

Every created convolutional and deconvolutional layer use a random-normal kernel initializer with standard deviation 0.01 and a L2 kernel regularizer with L2 0.001.

Once the network structure is defined, the optimizer and the cross-entropy lost is defined on the `optimize`(from line 116 to line 136) method using Adam optimizer. The network is trained using the `train_nn` (from line 140 to line 174) using keep probability 0.5 and learning rate 0.00001. To facilitate the loss value analysis, later on, every batch loss values are stored in an array, and the array is printed for each epoch. Examples after different epochs are provided in the example folder.