The ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) had \( \sim 1200 \) images for each of 1000 categories in 2012.
AlexNet on ILSVRC-2012

AlexNet revolutionized the state-of-the-art in object recognition at the time.

Figure 1: Top-5 error rates on ILSVRC image classification over Time

Source: http://paddlepaddle.org/docs/develop/book/03.image_classification/index.html
The Overall Architecture

Consists of 8 layers: 5 convolutional + 3 fully connected

The split (i.e. two pathways) in the image above are the split between two GPUs.

Inputs: RGB images with $224 \times 224 \times 3 = 150528$ values
Consider an input of size $W \times H \times C$ going to a convolutional layer $L$ with square kernel size $K$ and $M$ output maps (channels). Then $L$ has:

- There are $WHM$ neurons (output units) in $L$, one for each of the $WH$ “pixels” in the input and across the channels in the output $M$.
- There are $K^2CM$ weights. $K^2C$ per filter (the size of each “piece” of the input run through the filter), and $M$ filters in total (one per output channel).
- There are $WHK^2CM$ connections. A single kernel processes $K^2C$ values in the input; this occurs for each of the $WHM$ output units.
In contrast, a fully connected layer mapping $WHC$ inputs to $WHM$ outputs has far more weights and connections for small filter size $K$.

<table>
<thead>
<tr>
<th></th>
<th>Fully Connected</th>
<th>Convolutional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Neurons</td>
<td>$WHM$</td>
<td>$WHM$</td>
</tr>
<tr>
<td>Number of Weights</td>
<td>$W^2H^2CM$</td>
<td>$K^2CM$</td>
</tr>
<tr>
<td>Number of Connections</td>
<td>$W^2H^2CM$</td>
<td>$WHK^2CM$</td>
</tr>
</tbody>
</table>

Notice that the weight sharing (i.e. using the same kernel per channel across the output units) has decreased the number of weights considerably.

We will omit the bias weights for simplicity.
Let $N_i = WHM$, $P_i = K^2CM$, and $U_i = WHK^2CM$ be the number of output units, parameters (weights), and connections of layer $L_i$. Note that stride and max pooling each reduce the input $W$ and $H$ by the given factor.

- Layer $L_1$
  96 kernels (output channels) each of size $11 \times 11 \times 3$.
  Stride 4: the input $W$ and $H$ shrink by a factor of 4.
  Thus: $W = H = 55$, $C = 3$, $M = 96$, $K = 11$.
  - $N_1 = 55^2 \times 96 = 290,400$
  - $P_1 = 96 \times 11^2 \times 3 = 34,848$
  - $U_1 = 55^2 \times 11^2 \times 3 \times 96 = 105,415,200$
The Convolutional Layers (II)

- **Layer \( L_2 \):** 256 kernels each of size \( 5 \times 5 \times 48 \). (Max pooling: \( 55/2 = 27 \).) So \( N_2 = 27^2 \times 256 = 186,624; P_2 = 2(5^2 \times 48 \times 128) = 307,200; U_2 = 223,948,800/2 = 111,974,400. 

- **Layer \( L_3 \):** 384 kernels each of size \( 3 \times 3 \times 256 \). (Max pooling: \( 27/2 = 13 \).) So \( N_3 = 13^2 \times 384 = 64,896; P_3 = 3^2 \times 256 \times 384 = 884,736; U_3 = 13^2 \times 3^2 \times 256 \times 384 = 149,520,384. 

Note: layers 2, 4, & 5 are not connected to the preceding layer between GPUs; thus, one computes them separately and multiplies by 2. Recall: \( N_i = WHM, P_i = K^2 CM, U_i = WHK^2 CM \)
The Convolutional Layers (III)

- Layer $L_4$: 384 kernels each of size $3 \times 3 \times 192$. So $N_4 = 13^2 \times 384 = 64,869$; $P_4 = 2(3^2 \times 192^2) = 663,552$; $U_4 = 13^2 \times 3^2 \times 384^2/2 = 112,140,288$.

- Layer $L_5$: 256 kernels each of size $3 \times 3 \times 192$. So $N_5 = 13^2 \times 256 = 43,264$; $P_5 = 2(3^2 \times 192 \times 128) = 442,368$; $U_5 = 13^2 \times 3^2 \times 384 \times 256/2 = 74,760,192$.

Note: layers 2, 4, & 5 are not connected to the preceding layer between GPUs; thus, one computes them separately and multiplies by 2. Recall: $N_i = WHM$, $P_i = K^2CM$, $U_i = WHK^2CM$. 
The Fully Connected Layers

For the fully connected layers, again let \( N_i, P_i, \) and \( U_i \) be the number of output units, parameters (weights), and connections of layer \( L_i \).

- Layer \( L_6 \): 4096 units. (Max pooling: input \( 13/2 = 6 \).)  
  \[ N_6 = 4096; \quad P_6 = U_6 = 6 \times 6 \times 256 \times 4096 = 37,748,736 \]
- Layer \( L_7 \): \( N_7 = 4096 \) units. \( P_7 = U_7 = 4096 \times 4096 = 16,777,216 \)
- Layer \( L_8 \): \( N_8 = 1000 \) units. \( P_8 = U_8 = 4096 \times 1000 = 4,096,000 \)

Notice that the number of parameters is much larger for the dense layers than the convolutional ones.
## Summary of Results

<table>
<thead>
<tr>
<th>Layer</th>
<th>Units</th>
<th>Weights</th>
<th>Connections</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$ (Conv)</td>
<td>290,400</td>
<td>34,848</td>
<td>105,415,200</td>
</tr>
<tr>
<td>$L_2$ (Conv)</td>
<td>186,624</td>
<td>307,200</td>
<td>111,974,400</td>
</tr>
<tr>
<td>$L_3$ (Conv)</td>
<td>64,896</td>
<td>884,736</td>
<td>149,520,384</td>
</tr>
<tr>
<td>$L_4$ (Conv)</td>
<td>64,869</td>
<td>663,552</td>
<td>112,140,288</td>
</tr>
<tr>
<td>$L_5$ (Conv)</td>
<td>43,264</td>
<td>442,368</td>
<td>74,760,192</td>
</tr>
<tr>
<td>$L_6$ (Dense)</td>
<td>4096</td>
<td>37,748,736</td>
<td>37,748,736</td>
</tr>
<tr>
<td>$L_7$ (Dense)</td>
<td>4096</td>
<td>16,777,216</td>
<td>16,777,216</td>
</tr>
<tr>
<td>$L_8$ (Dense)</td>
<td>1000</td>
<td>4,096,000</td>
<td>4,096,000</td>
</tr>
<tr>
<td>Conv Subtotal</td>
<td>650,080</td>
<td>2,332,704</td>
<td>553,810,464</td>
</tr>
<tr>
<td>Dense Subtotal</td>
<td>9192</td>
<td>58,621,952</td>
<td>58,621,952</td>
</tr>
<tr>
<td>Total</td>
<td>659,272</td>
<td>60,954,656</td>
<td>612,432,416</td>
</tr>
</tbody>
</table>
Conclusion

Overall, AlexNet has about 660K units, 61M parameters, and over 600M connections.

Notice: the convolutional layers comprise most of the units and connections, but the fully connected layers are responsible for most of the weights.

More modern networks can do better with fewer parameters (e.g. GoogLeNet).
Further Reading

The original paper:


It’s possible to reduce the number of parameters of AlexNet by 9x without losing accuracy:

• Han, Song, et al. “Learning both weights and connections for efficient neural network.” Advances in neural information processing systems. 2015.