**ALEXNET**

* Earlier datasets such as NORB and CIFAR-10/100 contained tens of thousands of labeled images which were sufficient to do simple tasks but inadequate for recognizing images.
* Recently ,large datasets like **Imagenet(15M+ images)** and **LabelMe** using them we made even more complex models but the problem remaining under-specified. CNN’s use convolutional filters(kernels) which slide over the image this decreases the trainable parameters compared to a fully connected network which makes them easier to train.
* Despite CNN’s efficiency , the computational costs on highly resolved images is really expensive. However, using GPU’s and optimized 2D convolutional implementations we train CNN’s on huge datasets like Imagenet without severe overfitting.
* However, training is constrained by GPU’s memory and training time .But can be further improved by faster GPU’s and larger datasets.
* Starting in 2010, the **ImageNet Large-Scale Visual Recognition** **Challenge (ILSVRC**) was introduced as part of the Pascal Visual Object Challenge. This annual competition uses a subset of ImageNet containing: 1.2 million training images,50,000 validation images and 150,000 testing images. Each category in ILSVRC has around 1000 images, and there are 1000 categories in total.
* Since ImageNet contains **variable-resolution images**, but CNNs require a **fixed input size**, the images were **pre-processed** them like **Down-sampling to 256 × 256 resolution** and **Pixel Normalization**(trained directly on **centered raw RGB values** of pixels)

**ARCHITECTURE OF ALEXNET:**

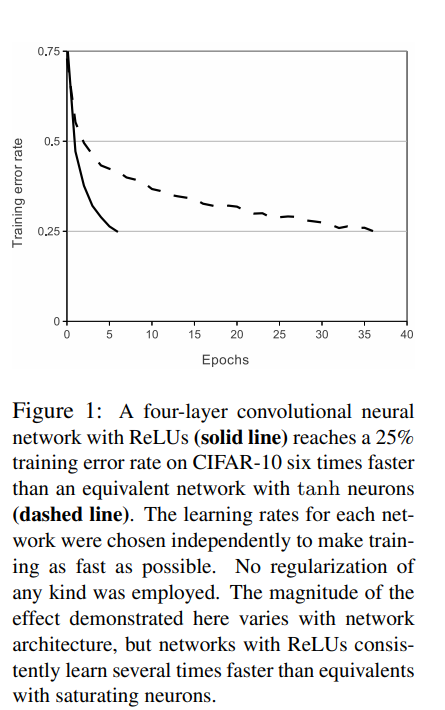
Some of the unusual features of the architecture being :

1. **Rectified Linear Units(ReLU) Non-Linearity:**

* Traditional neuron activation functions, such as **tanh** and **sigmoid**, suffer from saturation(when neurons in deep layers stop learning), making gradient descent slow(model learns slowly or stops learning).
* Instead, **Rectified Linear Units (ReLUs)**, defined as **f(x) = max(0, x)**, enable faster training.
* Deep CNNs with ReLUs train several times faster than those using tanh, as demonstrated on CIFAR-10, allowing experimentation with larger models.

1. **Training on Multiple GPUs:**

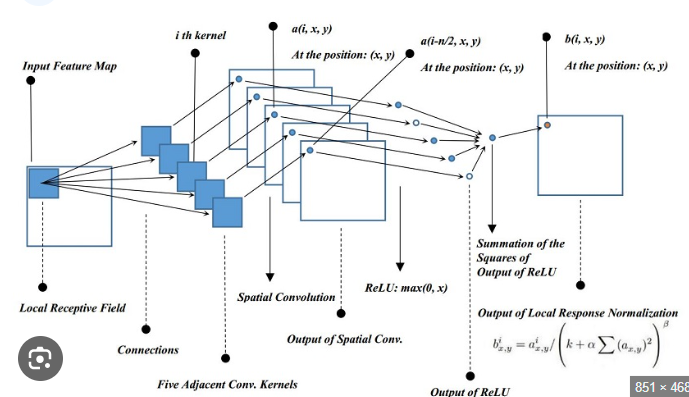
* Due to the **3GB memory limit** of a single GTX 580 GPU, training large networks on one GPU is not possible. Since 1.2 million training examples require larger networks, the model is **split across two GPUs**.
* GPUs can directly share memory without passing through the host system. The model is divided so that half of the kernels (neurons) are on each GPU i.e. Layer3 kernels receive input from all layer 2 maps and Layer4 kernels only receive input from layer 3 maps on the same GPU.

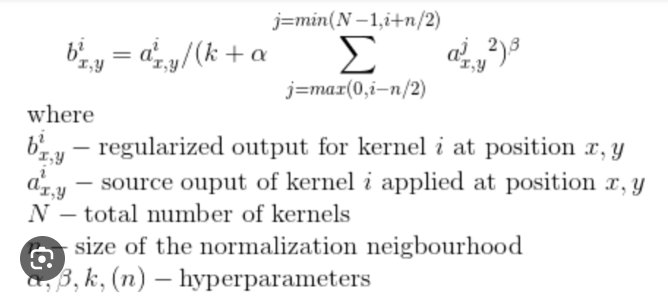


* This method is similar to **columnar CNNs** but with **interdependent** columns. The two-GPU network trains slightly faster than the single GPU network increasing the efficiency.

1. **Local Response Normalization(LRN):**

* ReLU neurons don’t require input normalization to prevent saturation, but local response normalization (LRN) helps improve generalization(performs better on unseen examples).



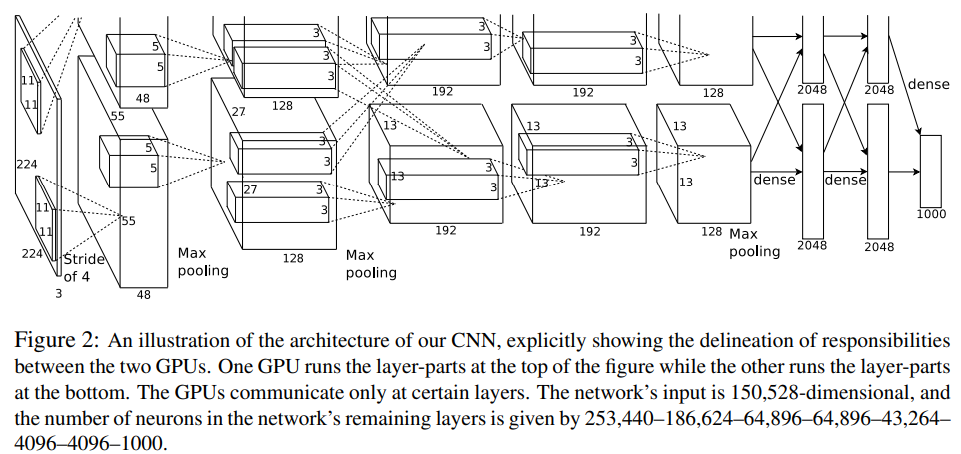


* It **creates competition among neurons**, similar to **lateral inhibition** in biological neurons, preventing some neurons from dominating.
* It’s like a **brightness normalization** (adjusting the intensity of an image / feature map) rather than contrast normalization. It also reduces the model’s error rate.

1. **Overlapping Pooling:**

* Pooling layers in CNNs summarize the outputs of nearby neurons in the same feature map. Traditionally, pooling is done in non-overlapping regions like (s = 2, z = 2).
* However, **overlapping pooling** is used in this network, with s = 2 and z = 3. This approach slightly reduces error rates (**top-1 by 0.4% and top-5 by 0.3%**) compared to non-overlapping pooling while maintaining the same output dimensions.
* Overlapping pooling helps reduce **overfitting** during training.

**OVERALL ARCHITECTURE:**

****

* The network consists of 8 layers in which 5 are convolutional layers and 3 are fully-connected layers and the final FC layer goes through a 1000-way softmax layers predicting the class probabilities . This model maximizes the **multinomial logistic** **regression** objective i.e. the correct one having the highest probability.
* The 2nd, 4th, and 5th convolutional layers connect only to maps on the same GPU. The 3rd convolutional layer connects to all maps in the 2nd layer. Fully-connected layers are fully connected to the previous layer.
* Response normalization is applied after the 1st and 2nd convolutional layers. Max-pooling follows response-normalization layers and the 5th convolutional layer. ReLU activation is applied after every convolutional and fully-connected layer**.**

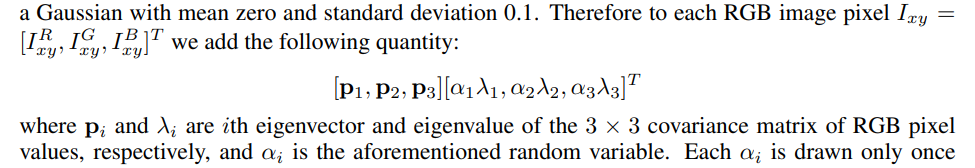
**REDUCING OVERFITTING:**

The neural network has 60 million parameters, but the 1000-class ILSVRC dataset provides only 10 bits of constraint per training example, which is insufficient to prevent overfitting.

**DATA AUGMENTATION:**

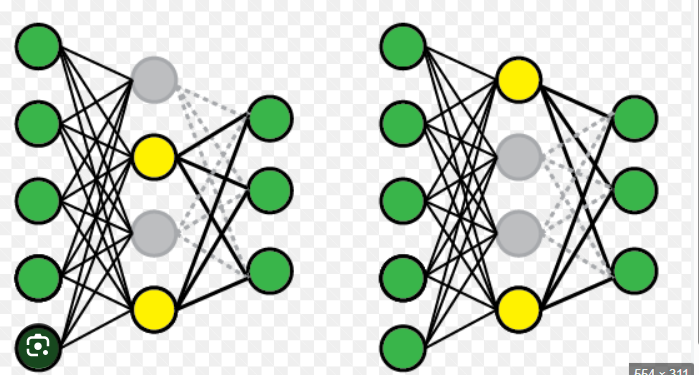
To reduce the overfitting, the dataset is artificially enlarged using two forms of data augmentation, both of which require minimal computation and are generated on the CPU while the GPU trains.

* Random **224 × 224 patches** (and their horizontal reflections) are extracted from **256 × 256 images** for training. This increases the dataset size **2048×**, helping prevent overfitting.
* Principal Component Analysis (**PCA**) is performed on **RGB pixel values** across the ImageNet training set.Perturbations are added to each pixel using principal components scaled by eigenvalues and a **random Gaussian variable** (mean = 0, std = 0.1). The same perturbation is applied across all pixels in an image but **recomputed** when the image is used again in training.



* This scheme approximately captures an important property of natural images, namely, that **object identity is invariant** to changes in the **intensity** and **colour of the illumination**. This scheme reduces the top-1 error rate by over 1%.

**DROPOUT:**

* Dropout is a regularization technique that reduces overfitting by randomly dropping (setting to zero) the output of each hidden neuron with (probability 0.5) during training.
* Each input sees a different NN architecture due to randomly dropped neurons. This prevents neurons from **over-relying** on specific co-adaptations and forces each neuron to learn **more general** features. Training time would get doubled.
* We use dropout in the first two fully-connected layers of Figure 2. 

**RESULTS:**

