SqueezeNet

SqueezeNet is a deep convolutional neural network that was designed to be extremely small and fast while maintaining good accuracy.
It was introduced by researchers from DeepScale, UC Berkely and Standford in a 2016 paper titled: "Squeezenet: AlexNet level accuracy with 50x fewer parameters and <0.5 MB model size."
The title perfectly captures its achievements: it matched the accuracy of (then famous) AlexNet architecture on the ImageNet dataset, but with a model that was over 500 times smaller in size (less than 1MB compared to AlexNet's ~240MB)

Why it came into existence?

At the time, leading models like VGGNet and GoogleNet were becoming incredibly large and computationally expensive. This made them powerful for research but impractical for real-world applications on resource-constrained devices like Mobile phone, embeded IOT devices.

The core architecture "Fire Module"

SqueezeNet's efficiency doesnot come from complex math but from a clever and repeating building block called the "Fire Module". This module is built on three key design strategies:

1. Aggressively replace 3x3 filters with 1x1 filters. A 1x1 filter has 9 times fewer parameters than 3x3 filters.
2. Decrease the number of input channels to 3x3 filters. Before applying 3x3 filters, they "squeeze" the input using 1x1 filters. This reduces the depth (number of channels), hence fewer weights in subsequent 3x3 convolutions.
3. Downsample late in the Network. Pooling layers reduce spatial resolution (height x width). By delaying pooling, early layers preserve more spatial information, improving accuracy.

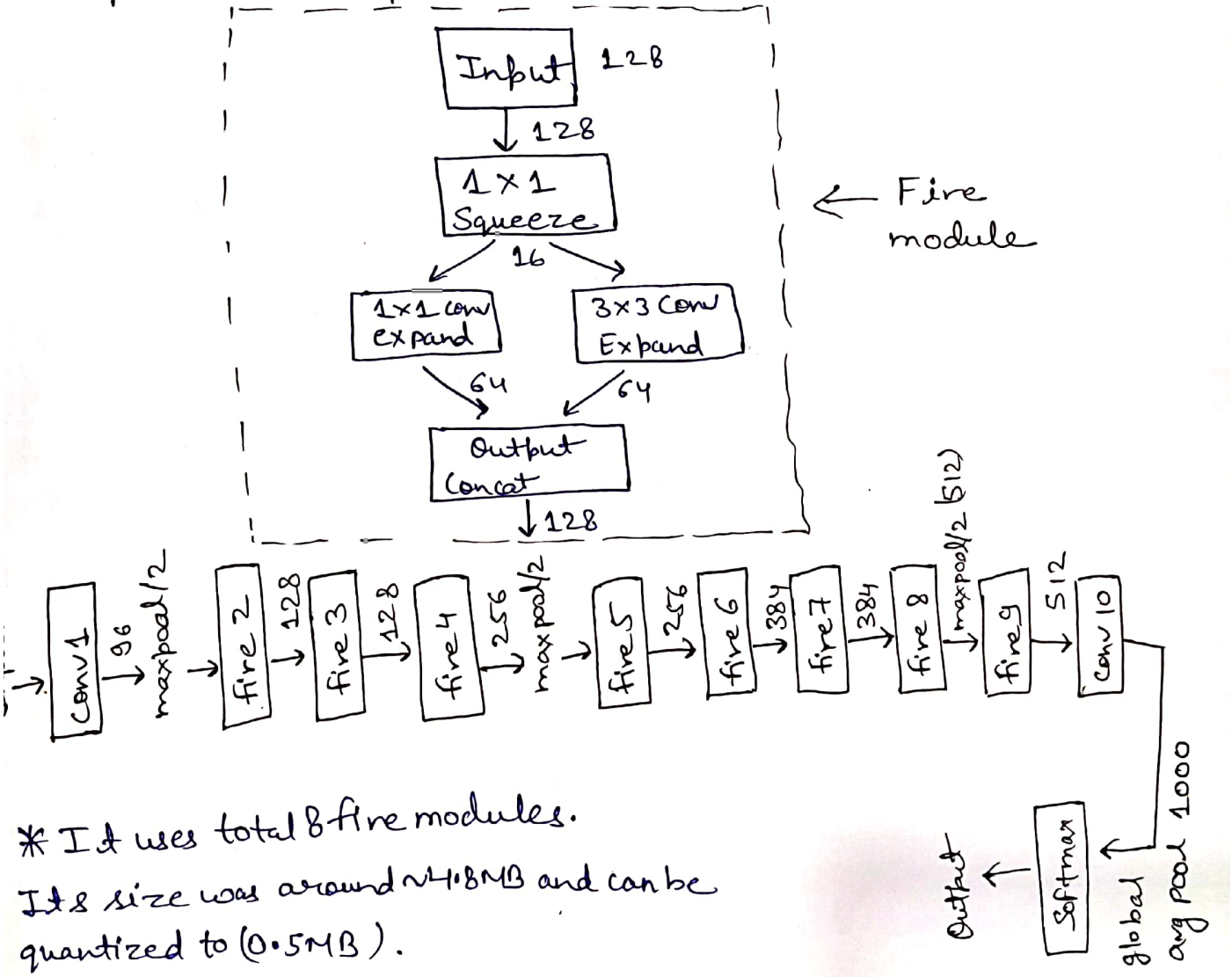
The Fire Module implements these strategies perfectly. It has two parts:

1. A squeeze layer: This layer uses only 1×1 convolution filters. It takes a large number of input channels and "squeezes" them down to a much smaller number. It outputs a fewer feature maps.
2. An expand layer: This layer takes the "squeezed" output and feeds it into two parallel convolutions, one with 1×1 filters and another with 3×3 filters. The results are then concatenated. 1×1 filter balances efficiency and 3×3 filter works for spatial coverage.

So the data flow is ⇒

Input → Squeeze (1×1 conv) → Expand (1×1 & 3×3 conv) → concatanate

This combination gives both parameter efficiency and representational power (multi-scale).



* It uses total 8 fire modules.

Its size was around ~4.8MB and can be quantized to (0.5MB).