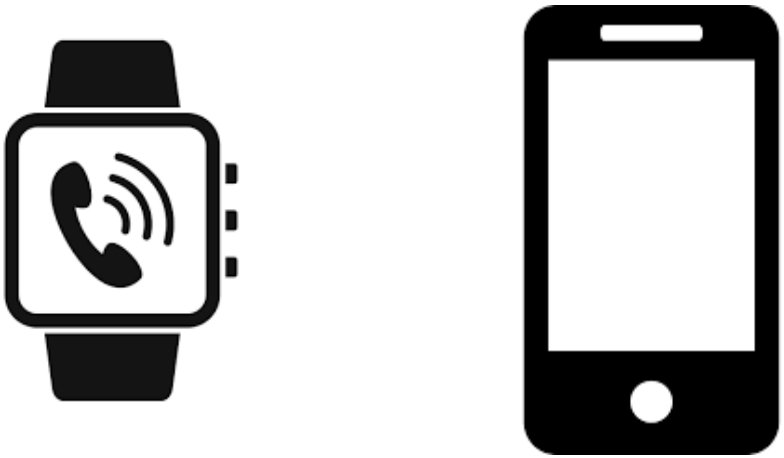
Adjusting NAS training towards more efficient networks

Miki Naimer
Yuval Goddard
Tal Van Dijk

Advisor: Matan Friedman

Goal

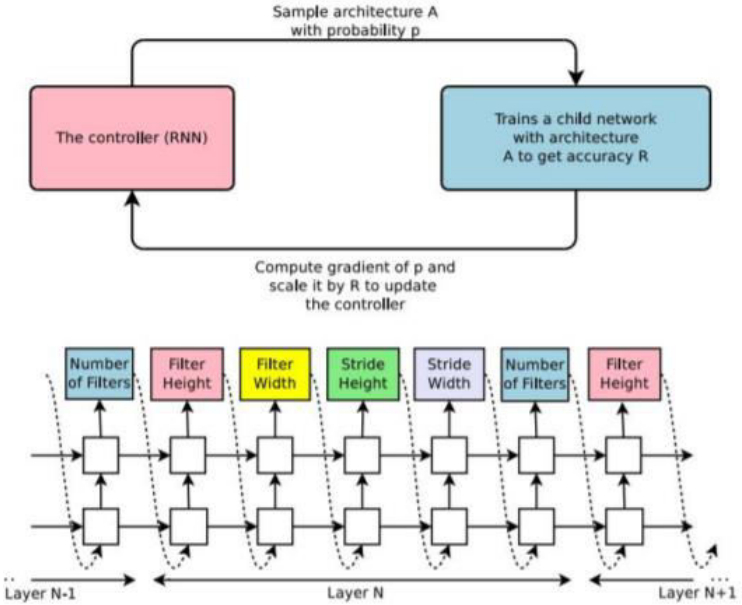
- AutoML for edge devices with limited resources



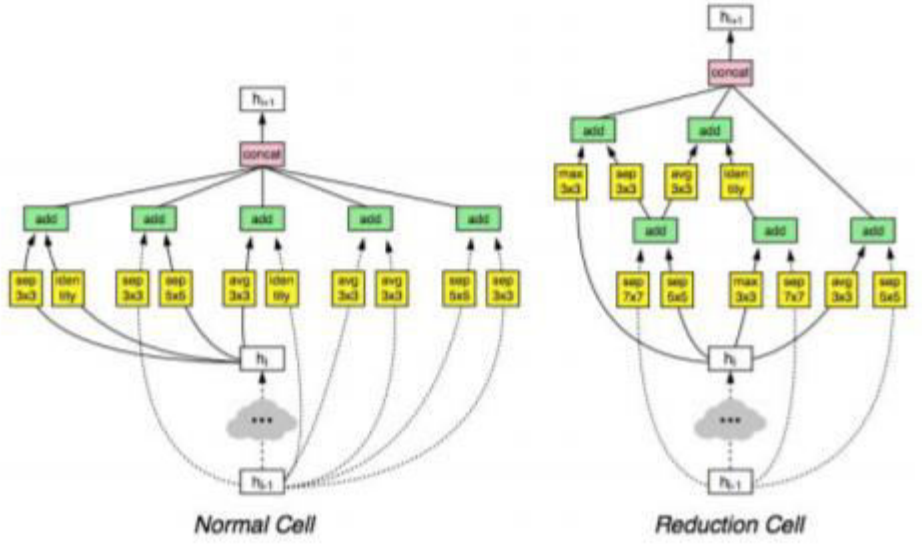
Approach

- Adjusting NAS training to search for network as small as possible
- Network size is measured by:
  - Number of parameters
  - Flops

NAS - Neural Architecture Search



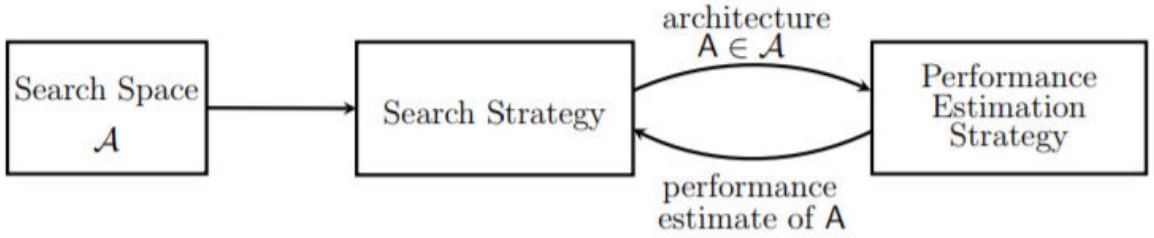
Zoph et al. 2016



Zoph et al. 2017

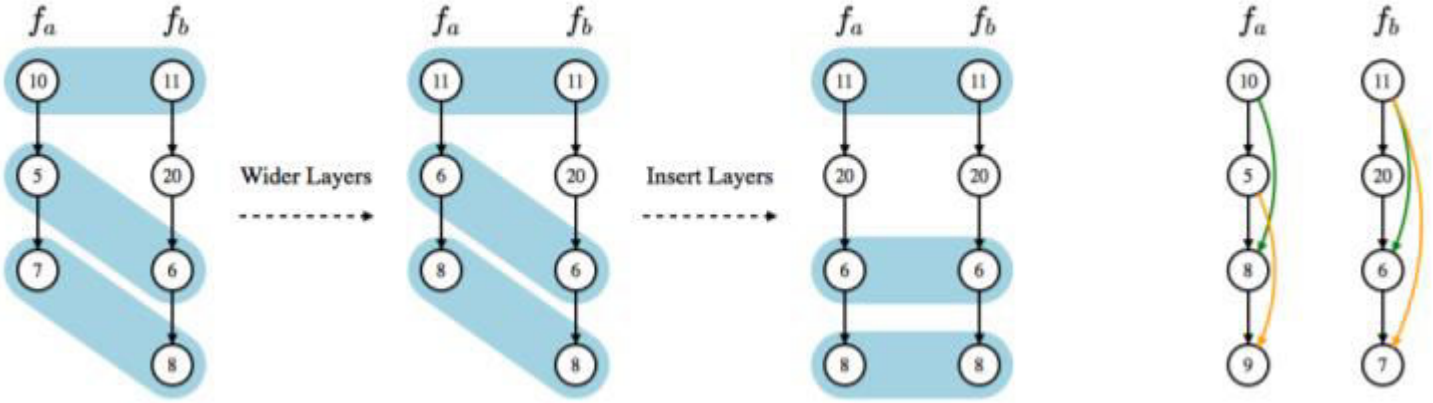
NAS - Neural Architecture Search

- Search Space - which architectures can be represented?
- Search Strategy:
  - Random search
  - Bayesian optimization
  - Evolutionary methods
  - Reinforcement learning (RL)
  - Gradient-based methods.
- Performance Estimation Strategy - training



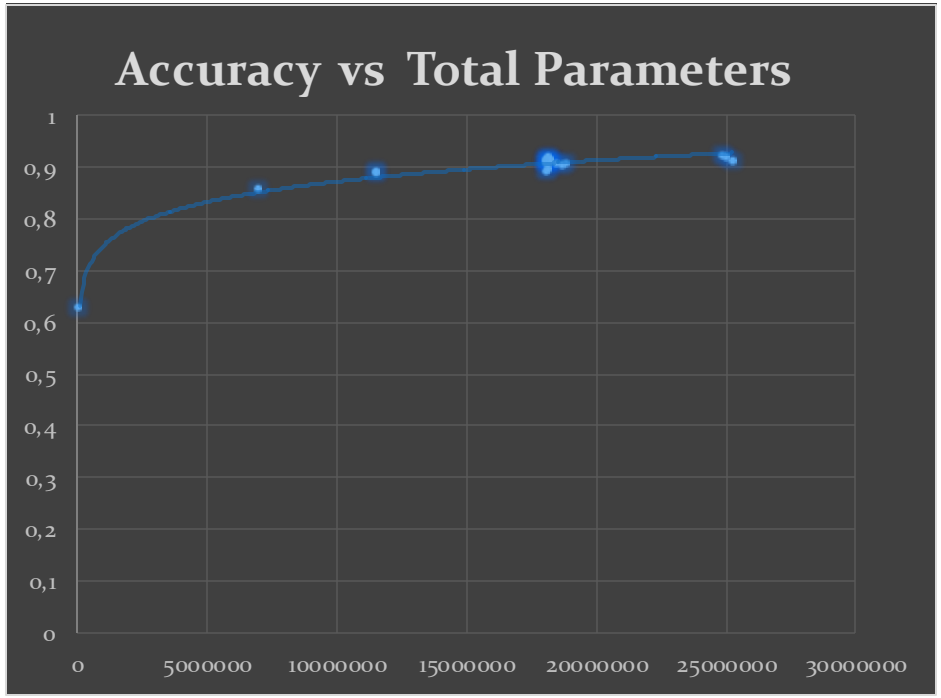
Autokeras

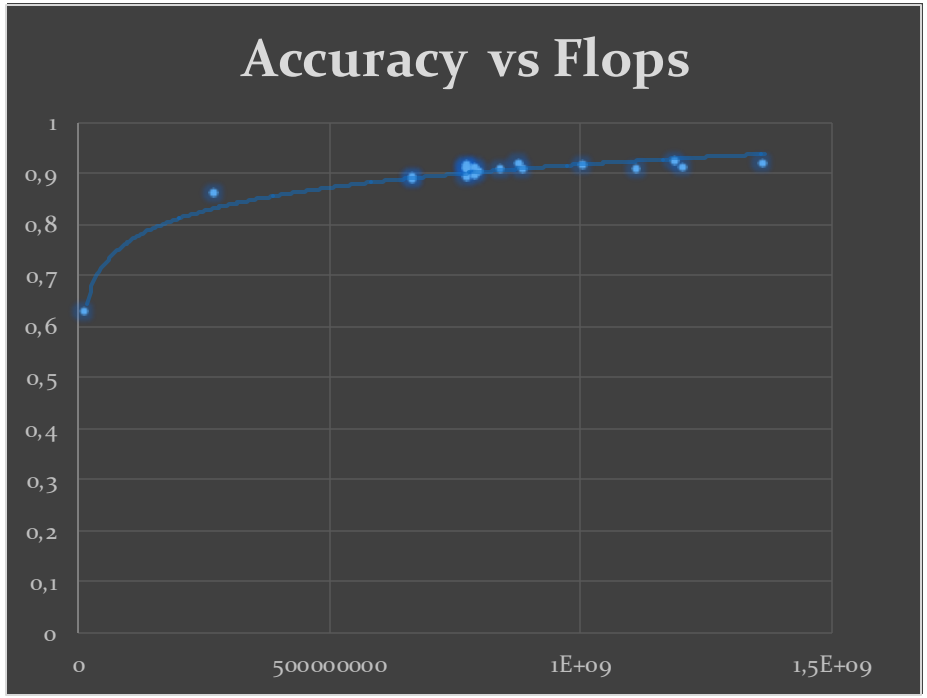
- Network morphism with Bayesian optimization
- Open source - https://autokeras.com/



AUTO KERAS

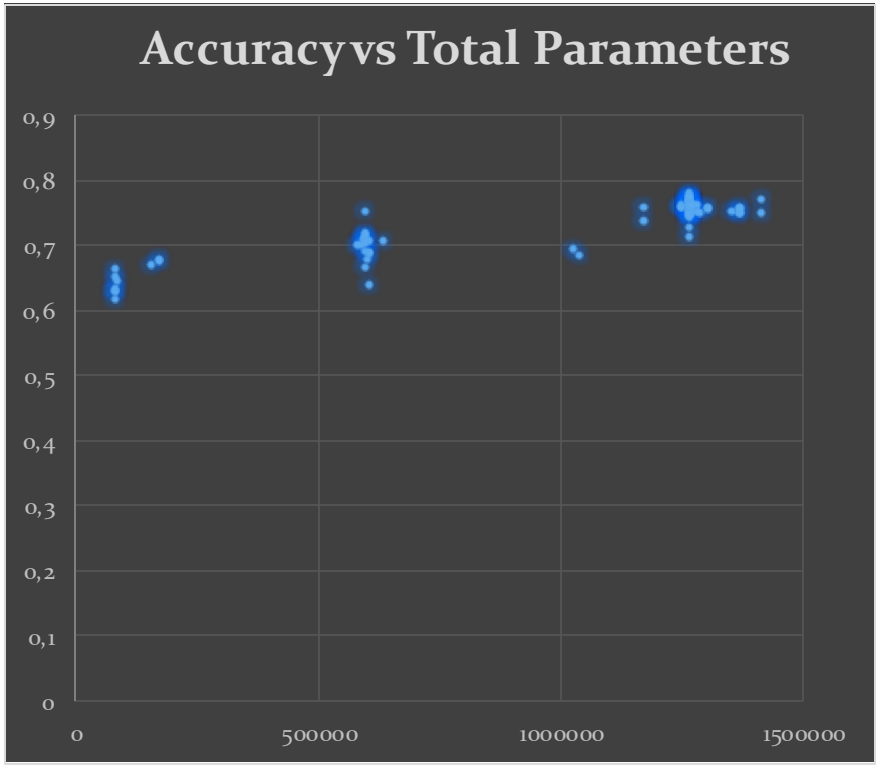
CIFAR 10 – Autokeras first results

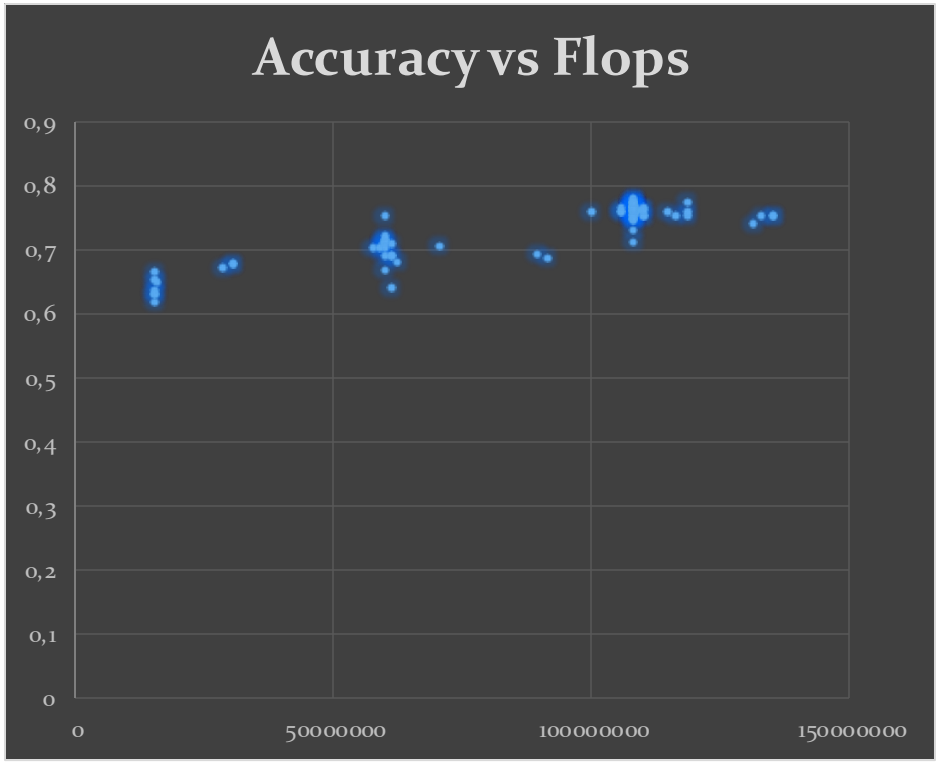




First try to decrease model size

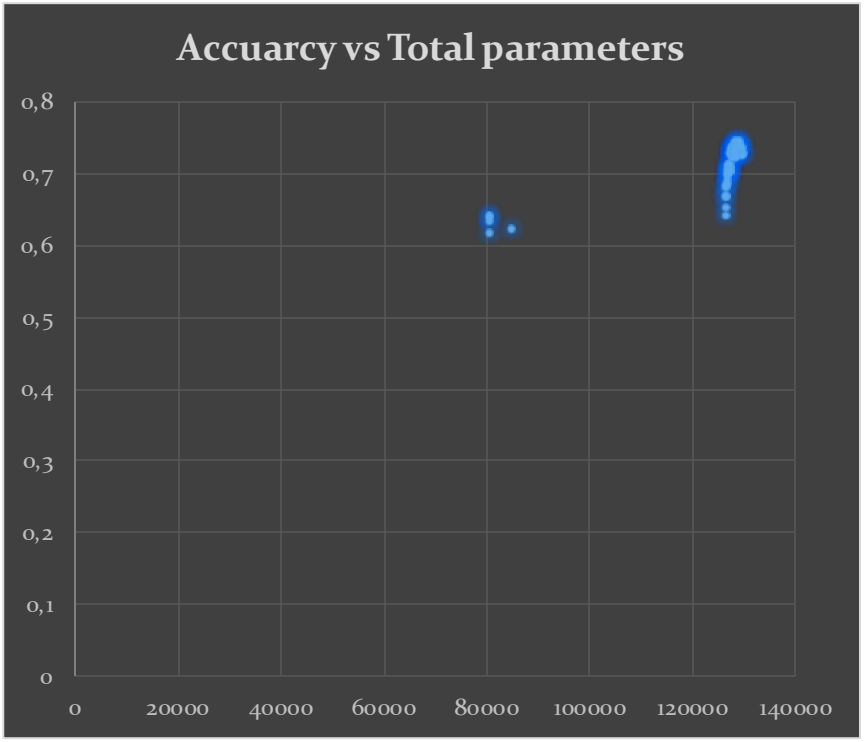
- Less GPU

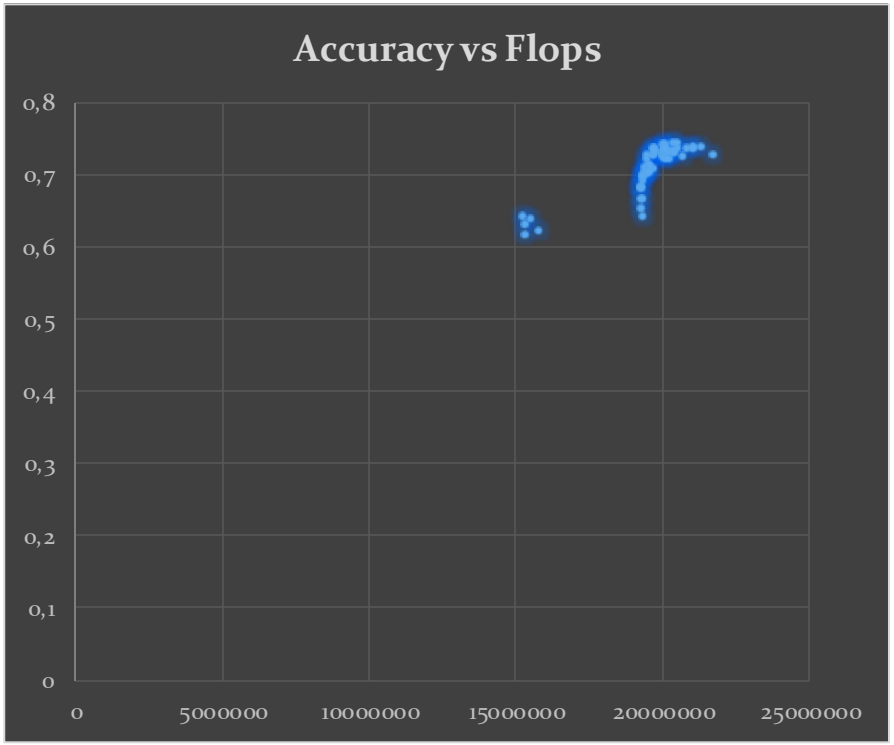




First try to decrease model size

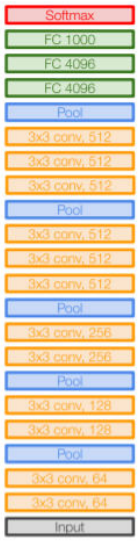
- Decrease max model size parameter



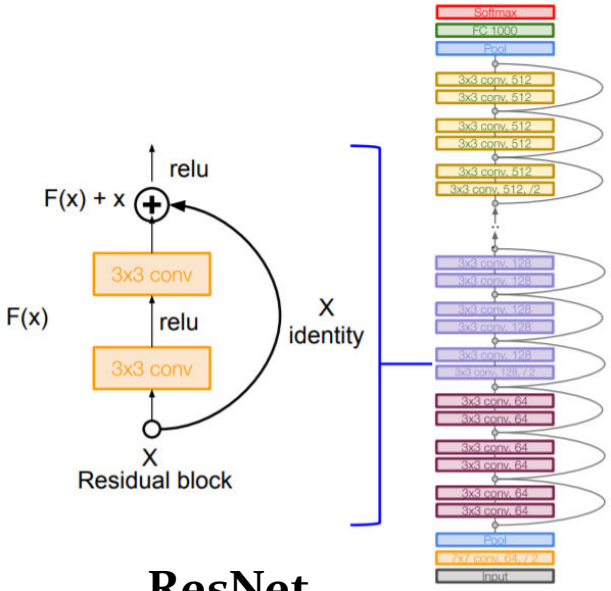


Autokeras base architectures

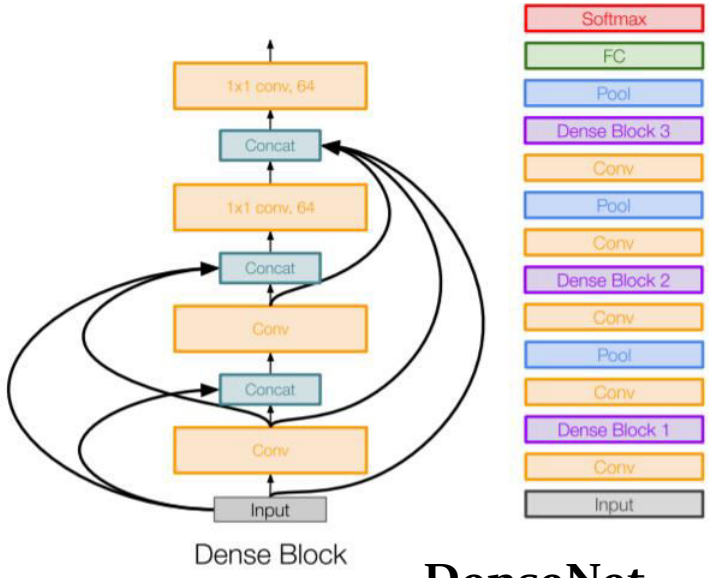
model	params	flops	accuracy
CNN	80720	15243510	0.6296
ResNet	11529354	667985920	0.8884
DenseNet	6974346	273725152	0.86



CNN



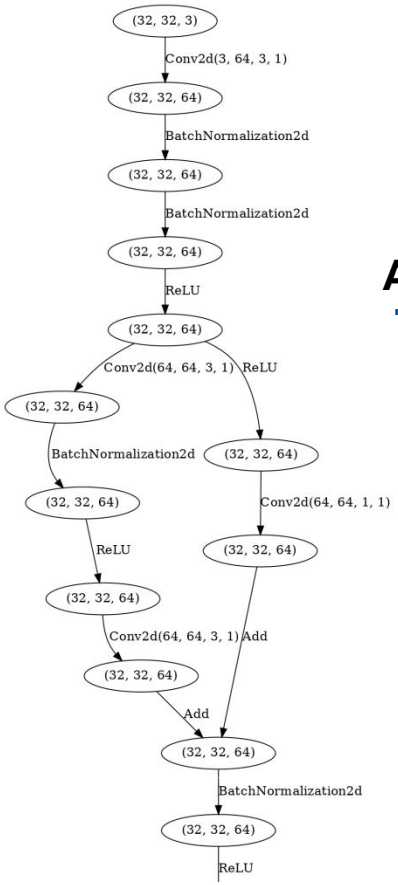
ResNet



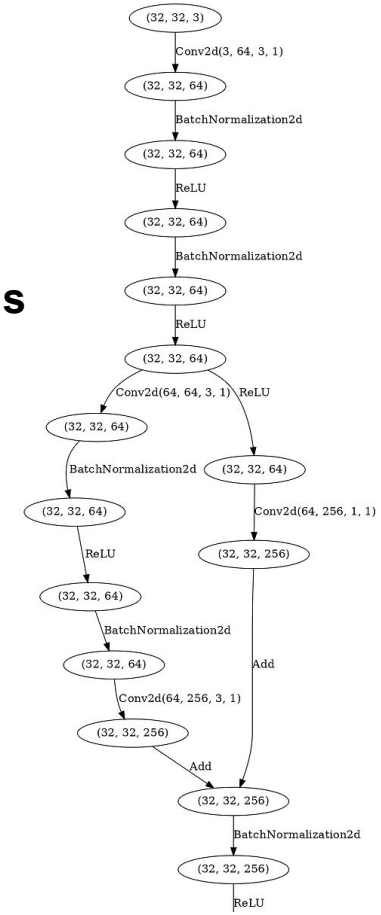
DenseNet

Deeper look at autokeras network morphism

- Network morphism operations:
  - Deep
  - Wide
  - Add
  - Concat
- Network morphism rate

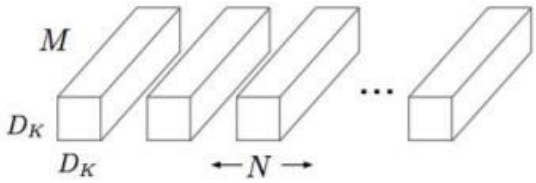


After 16 transforms

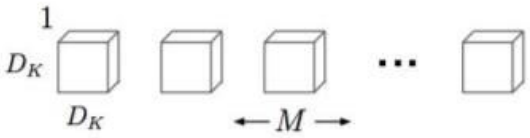


MobileNet

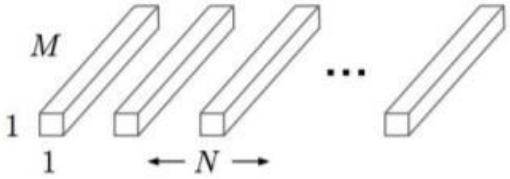
- Depthwise Convolutional Filters



(a) Standard Convolution Filters



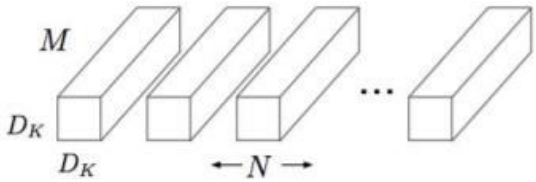
(b) Depthwise Convolutional Filters



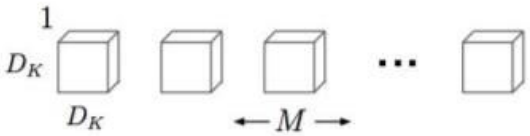
$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNet

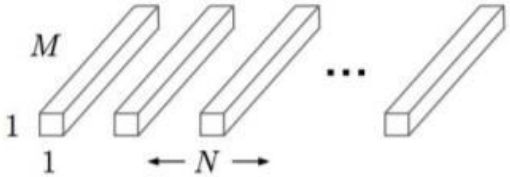
- Depthwise Convolutional Filters
- **Not supported in autokeras!**



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

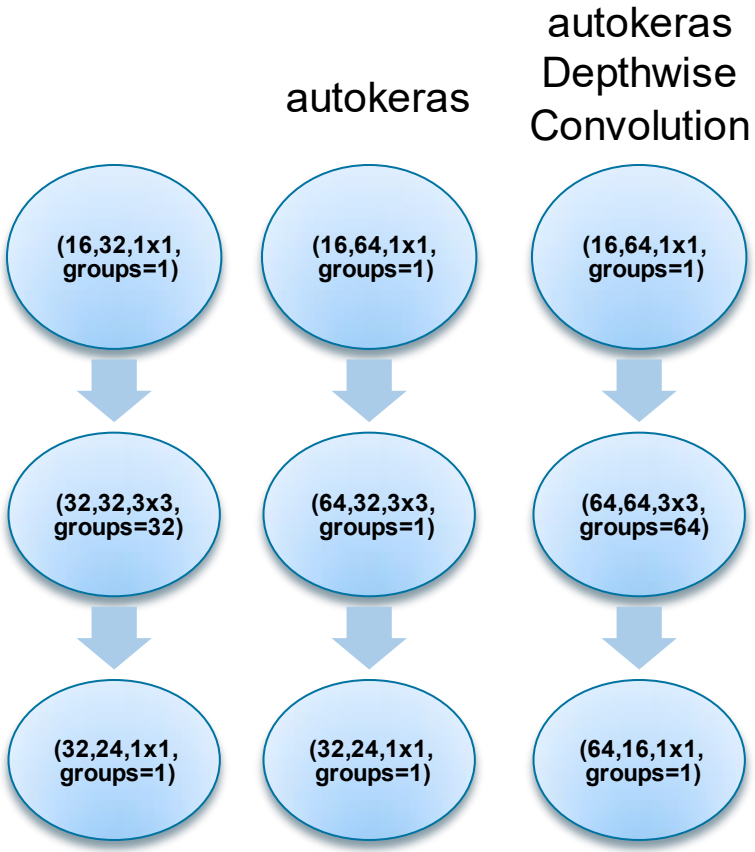
Depthwise Convolution support

- Autokeras used PyTorch
- PyTorch parameter groups
- Add groups parameter to autokeras layer

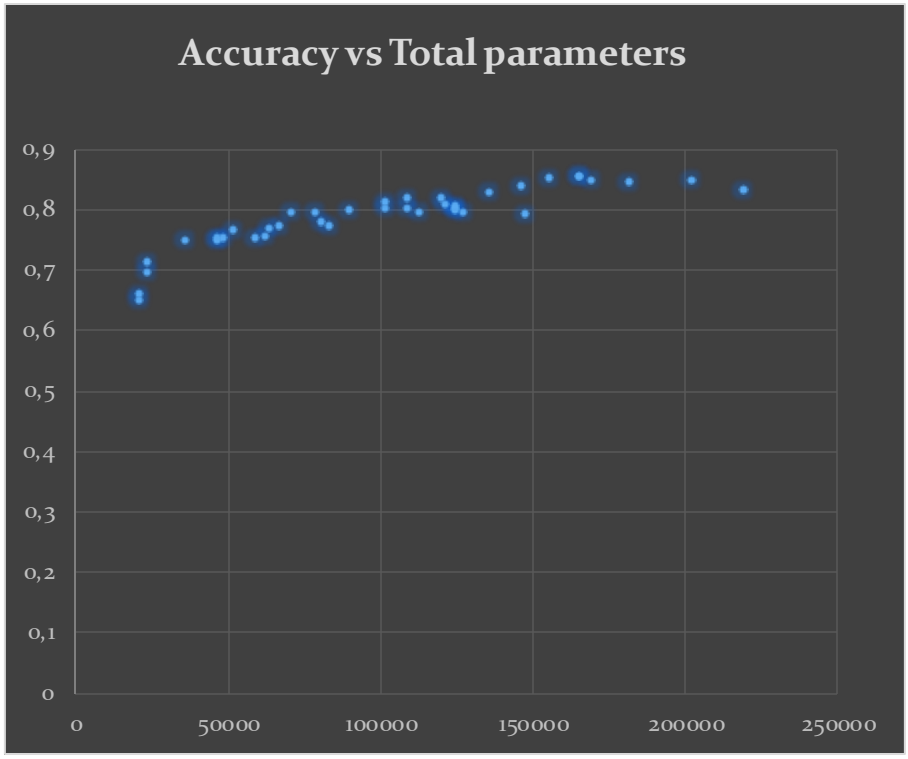
```
class StubConv(StubLayer):

    def __init__(self, input_channel, filters, kernel_size, stride=1,
                 padding=None, groups=None, output_node=None, input_node=None):
        super().__init__(input_node, output_node)
        self.input_channel = input_channel
        self.filters = filters
        self.kernel_size = kernel_size
        self.stride = stride
        self.padding = padding if padding is not None else int(self.kernel_size / 2)
        self.groups = groups if groups is not None else 1
```

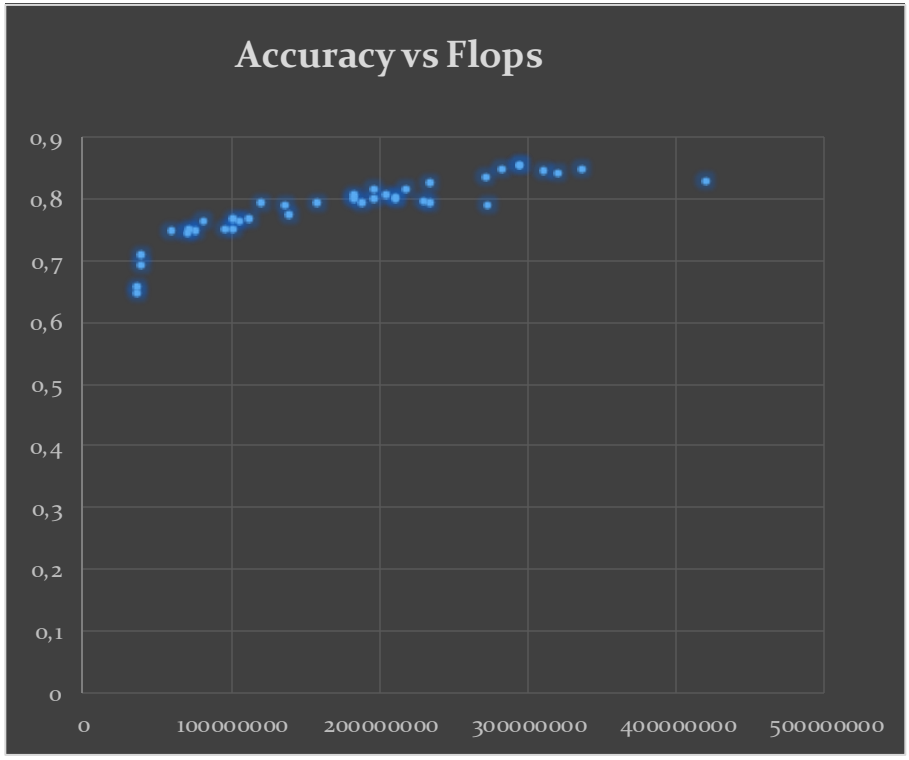
Depthwise Convolution support

- *Groups* constrains:
  - *groups* controls the connections between inputs and outputs
  - *in_channels* and *out_channels* must both be divisible by groups
  - At *groups*= *in_channels*, each input channel is convolved with its own set of filters, of size: $\left\lfloor \frac{out\_channels}{in\_channels} \right\rfloor$
- Weights



Mobilenet as base architectures





Summary & Future work

- Summary
  - NAS – a growing field of research
  - Autokeras base architectures
  - Depthwise Convolution in autokeras

- Future work
  - Effect of others base architectures
  - Effect of longer NAS session