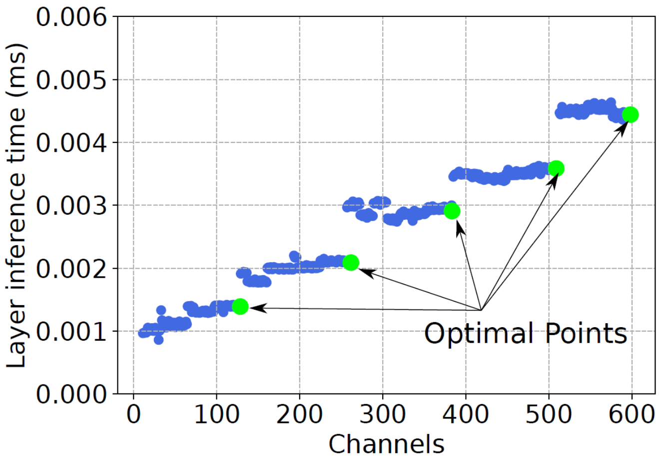
Staircase:
Distilling with Performance Enhanced Students for Hardware

Jack Turner[1], Elliot J. Crowley[1], Valentin Radu[1], José Cano[2], Amos Storkey[1], Michael O'Boyle[1]

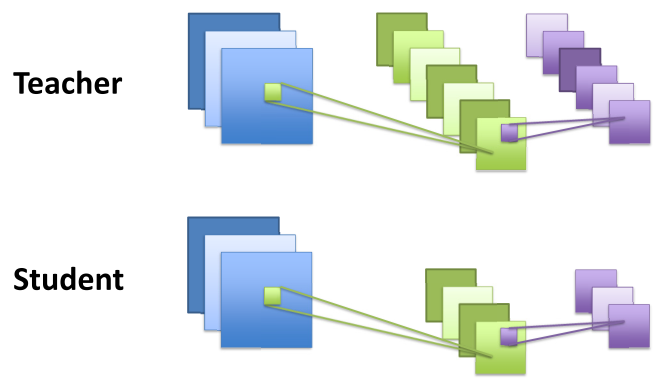
[1]School of Informatics, University of Edinburgh, UK - [2]School of Computing Science, University of Glasgow, UK

Motivation



- Inference time for a layer of **ResNet-34** vs #channels on **Intel Core i7**
- **Staircase pattern**: For a given inference time, the **green points** maximise the network's capacity (i.e. we get extra channels for no increase in time)

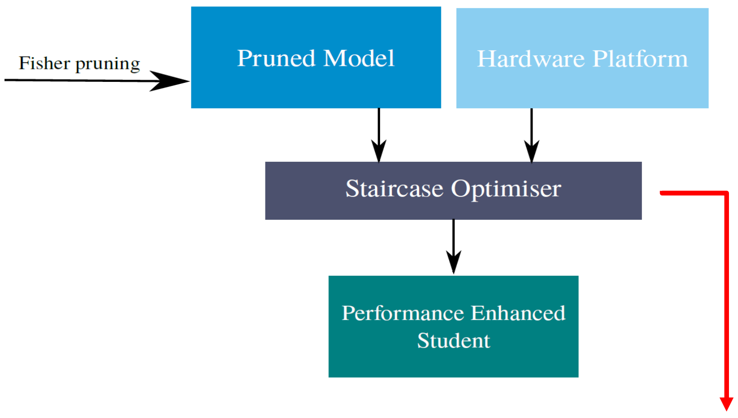
Model distillation



- **Idea**: propose a **novel channel pruning approach** that uses hardware behaviour to reshape networks
- A small **student network** is trained both on the **data** and **outputs** of a larger pre-trained **teacher network**

Discovery and optimisation pipeline (1)

- **Step 1:** Using channel saliency and empirical latency, design student

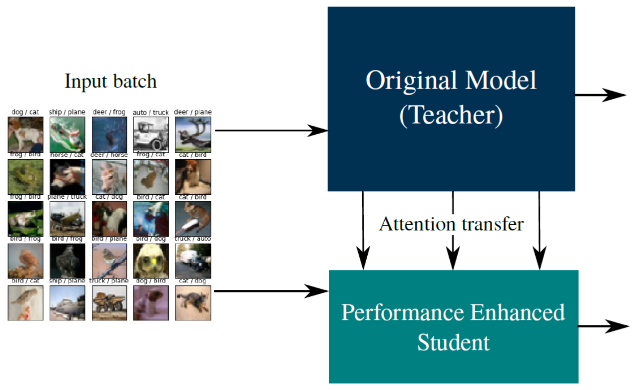


- **Student discovery algorithm**
  - **Starting**: base model, a Fisher-pruned reduction of the base model, and a target hardware platform
  - We **iterate over all prunable layers** in the base model and construct a set of optimal points
  - We then **adapt the pruned layer widths** to their nearest optimal point and return the resulting architecture

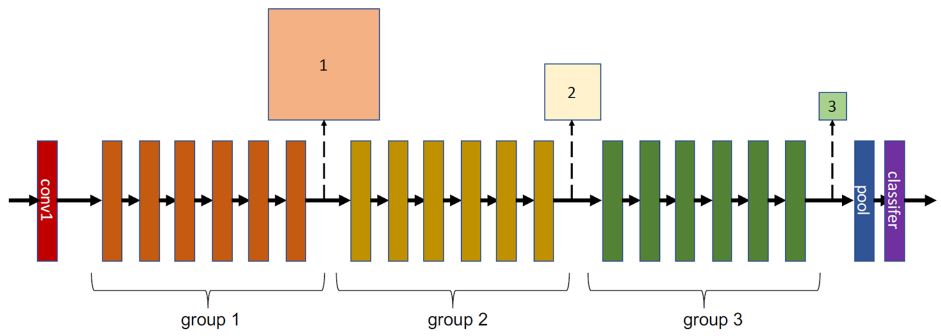
```
Target: Target hardware platform
BaseModel: Baseline pretrained model
FisherModel: Fisher-pruned BaseModel
student = Model()
for i, layer in enumerate(BaseModel) do
    fisher_width = FisherModel[i].layer_width
    base_layer = BaseModel[i]
    times = []
    for c in range(1 to base_layer.layer_width) do
        NewLayer = Conv(base_layer.in_channels, c)
        time = NewLayer(example_data)
        times.append(time)
    end
    opt_points = get_outliers(times)
    new_layer = Conv(base_layer.in_channels,
      nearest_neighbour(fisher_width, opt_points))
    student.append(new_layer)
end
```

Discovery and optimisation pipeline (2)

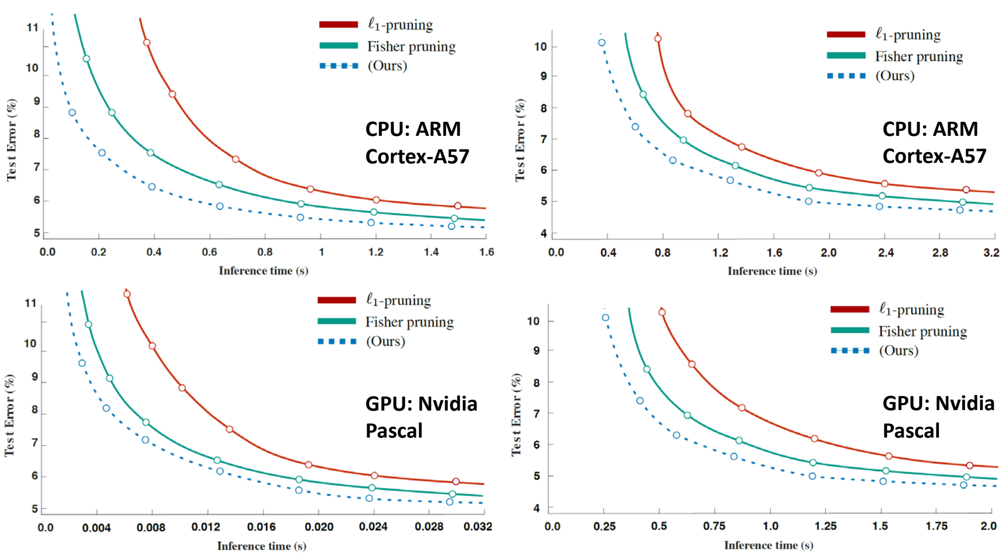
- **Step 2:** Train via attention transfer



- **Example:** block diagram of a WideResNet (attention maps:1, 2, 3)



Results: CIFAR-10



WideResNet-40-2 | **DenseNet-100**

Results: ImageNet

GPU: Nvidia Pascal

Network	Params	MACs	Top-1 Err	Top-5 Err	Speed	MACs/s
Baseline ResNet-34	21.3M	4.12G	21.84	5.71	0.122s	33.77G
Fisher-pruned ResNet-34	5.3M	1.44G	43.43	18.87	0.038s	37.89G
Our ResNet-34	6.8M	1.58G	31.29	11.16	0.040s	39.50G

Conclusions

- We have described a **simple method** for discovering **performance enhanced reductions** of baseline, large neural networks
- We have compared our technique to common pruning approaches, and demonstrated its **superiority** on both the **CIFAR-10** and **ImageNet** datasets for popular networks and hardware platforms

THE UNIVERSITY of EDINBURGH informatics

University of Glasgow | School of Computing Science

ARM Research Summit - Austin, USA
September 15-18, 2019

BONSEYES ARTIFICIAL INTELLIGENCE MARKETPLACE