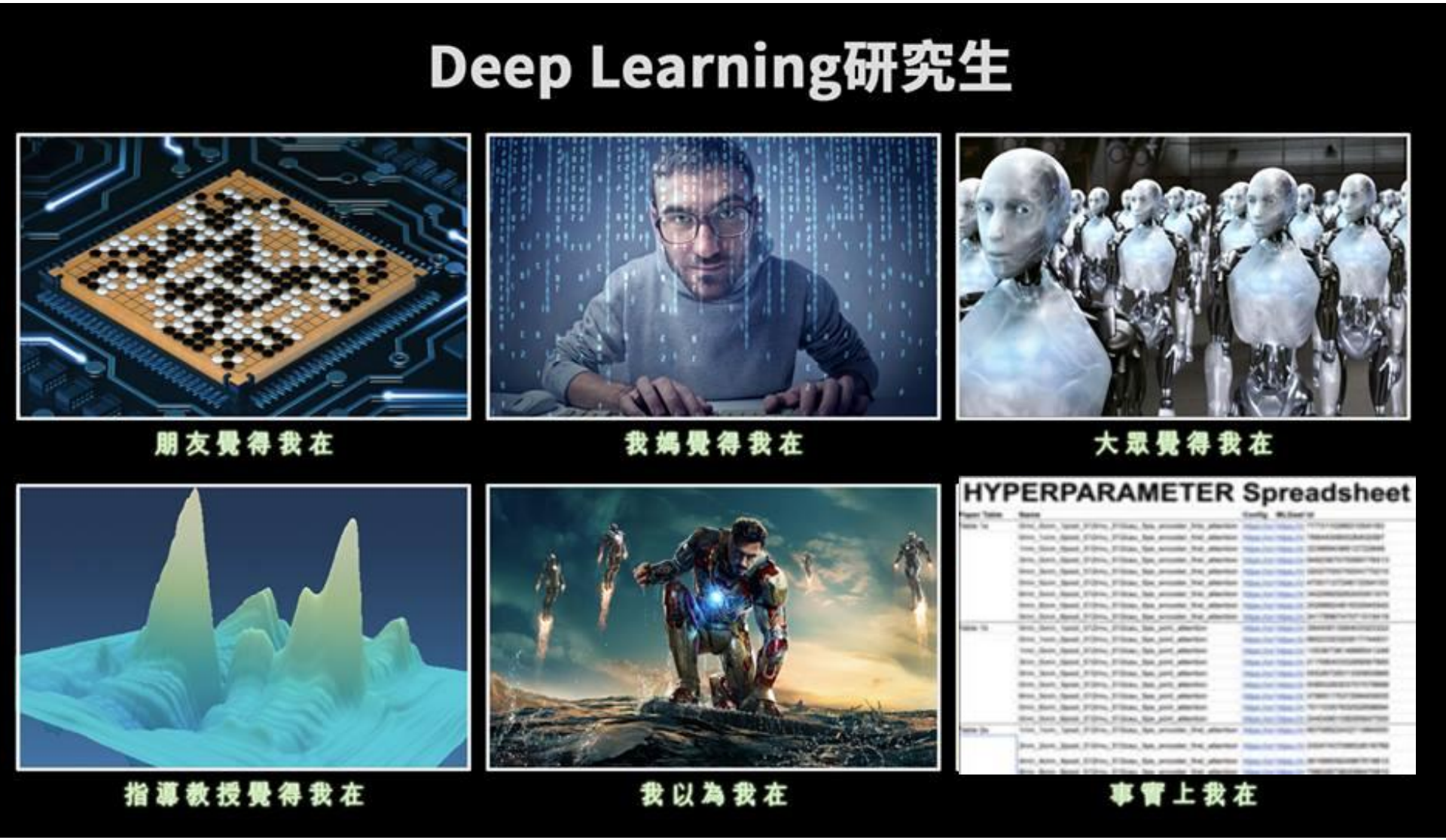
Automatically Determining Hyperparameters

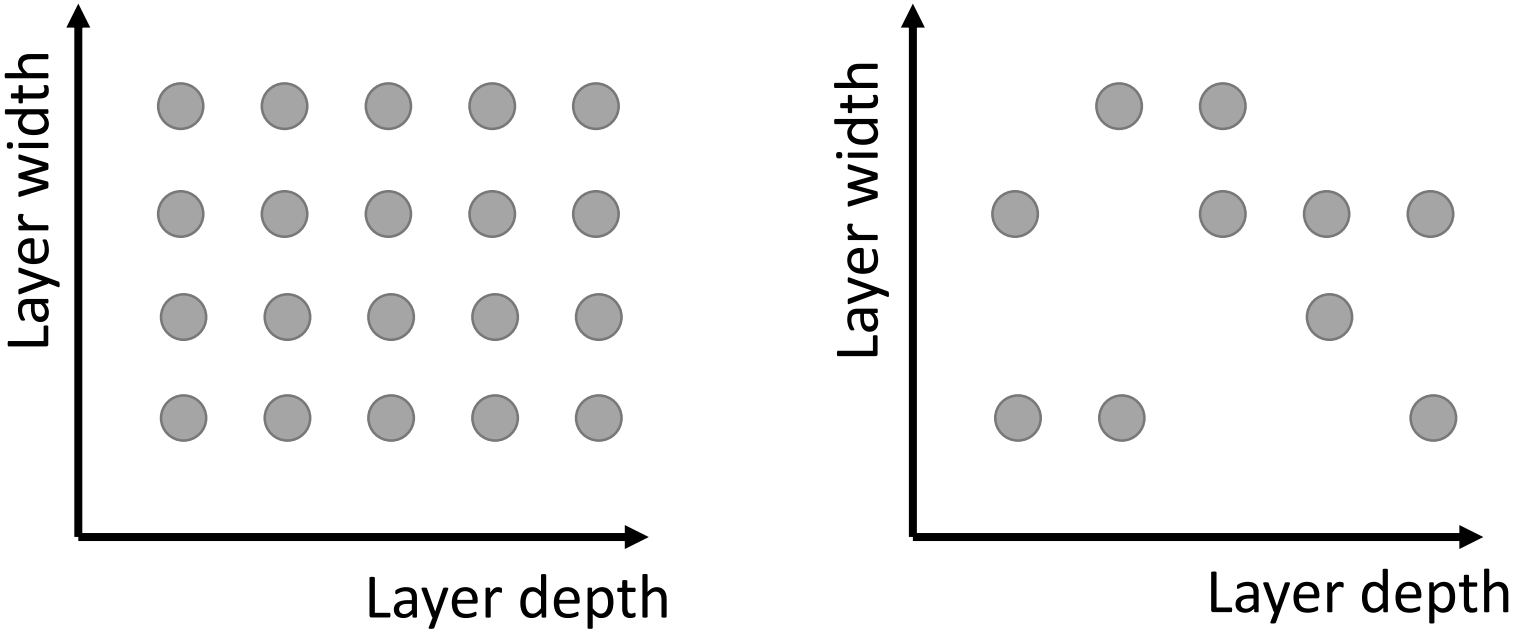
Source of iamge: https://medium.com/intuitionmachine/the-brute-force-method-of-deep-learning-innovation-58b497323ae5 (Denny Britz's graphic)



感謝 沈昇勳 同學提供圖檔

http://www.deeplearningbook.org/contents/guidelines.html

Grid Search v.s. Random Search



Assumption: top K results are good enough

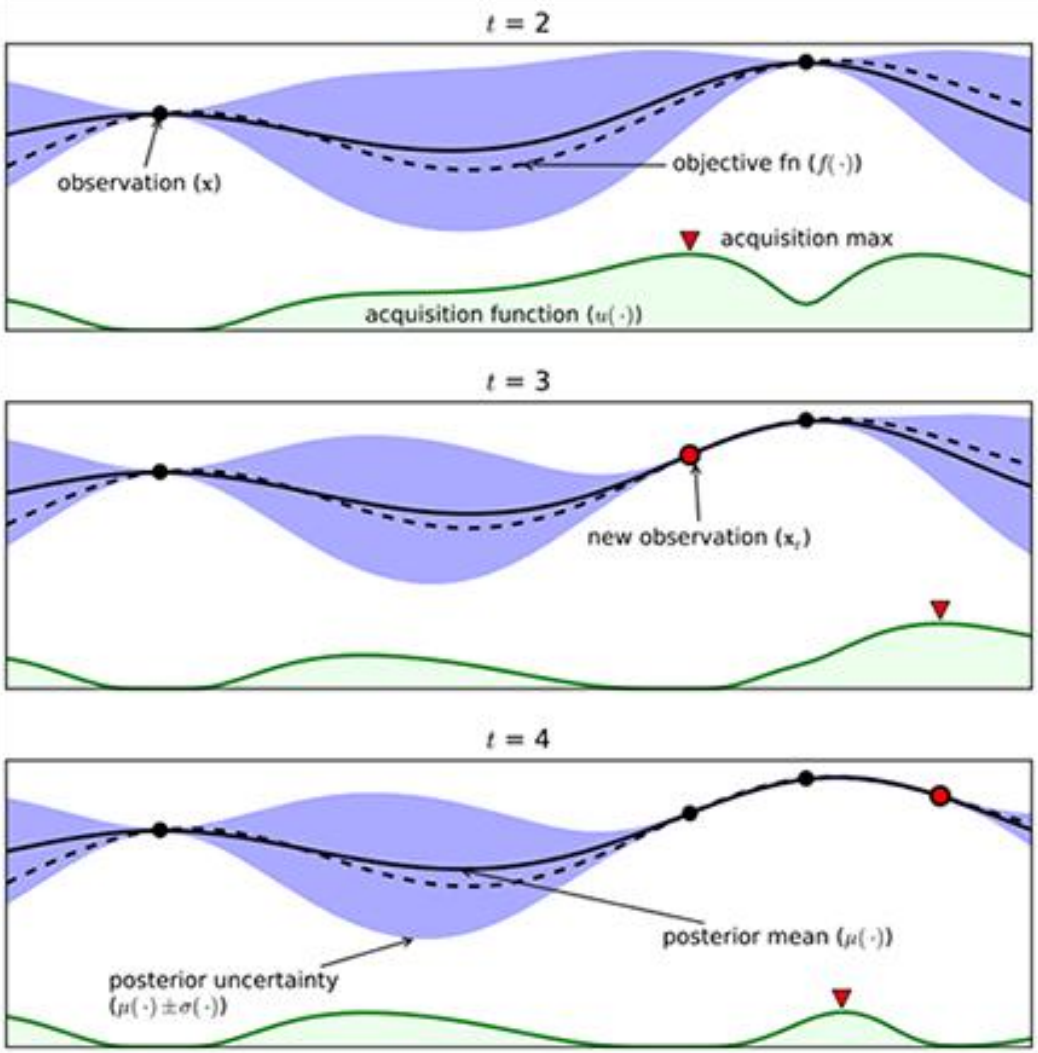
If there are N points, probability K/N that your sample is in top K

Sample x times: $1 - (1 - K/N)^x > 90\%$

If N = 1000, K = 10 → x = 230

K = 100 → x = 22

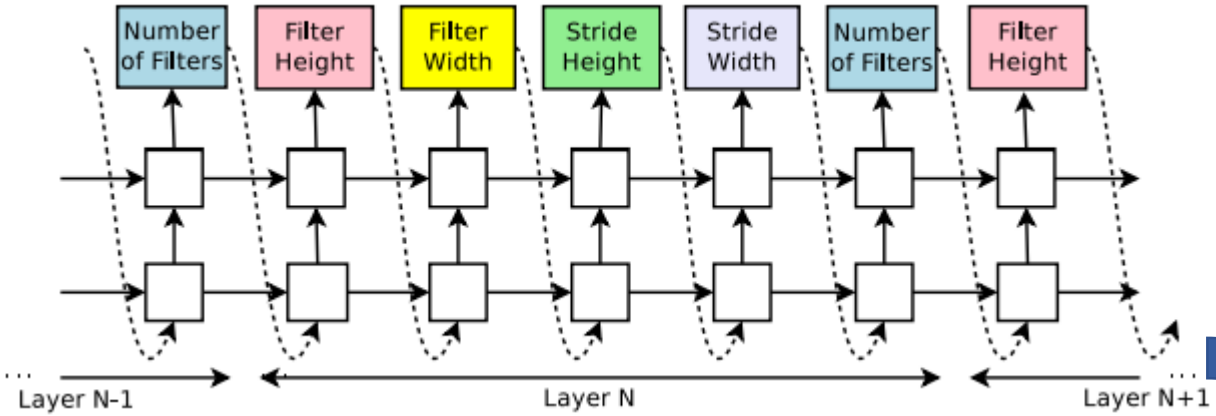
Model-based Hyperparameter Optimization



https://cloud.google.com/blog/big-data/2017/08/hyperparameter-tuning-in-cloud-machine-learning-engine-using-bayesian-optimization

Reinforcement Learning

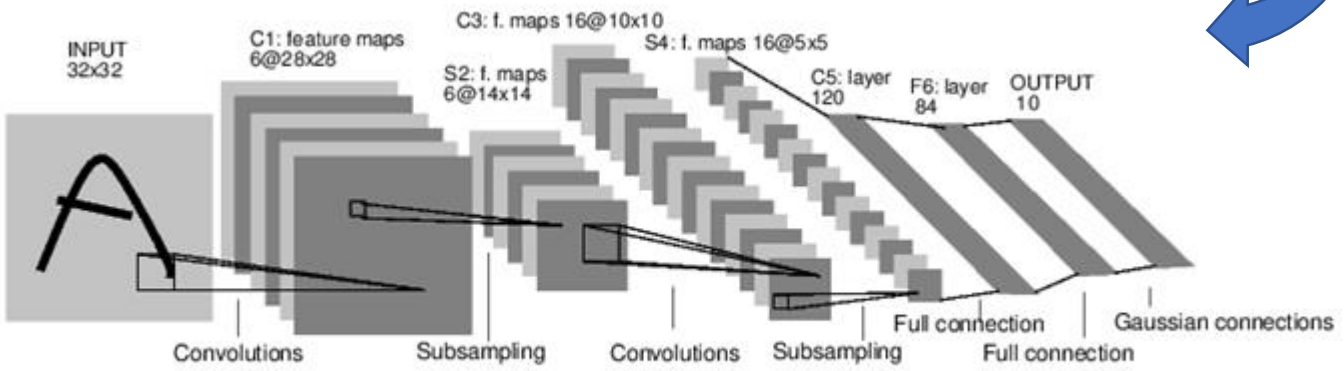
One kind of meta learning (or learn to learn)



Accuracy as reward

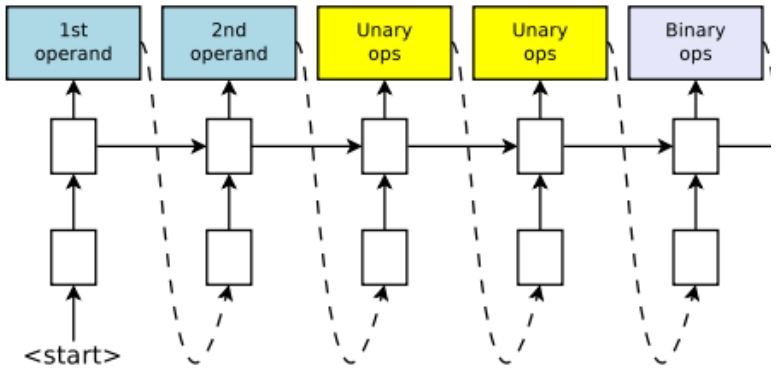
Design a network

Train the network

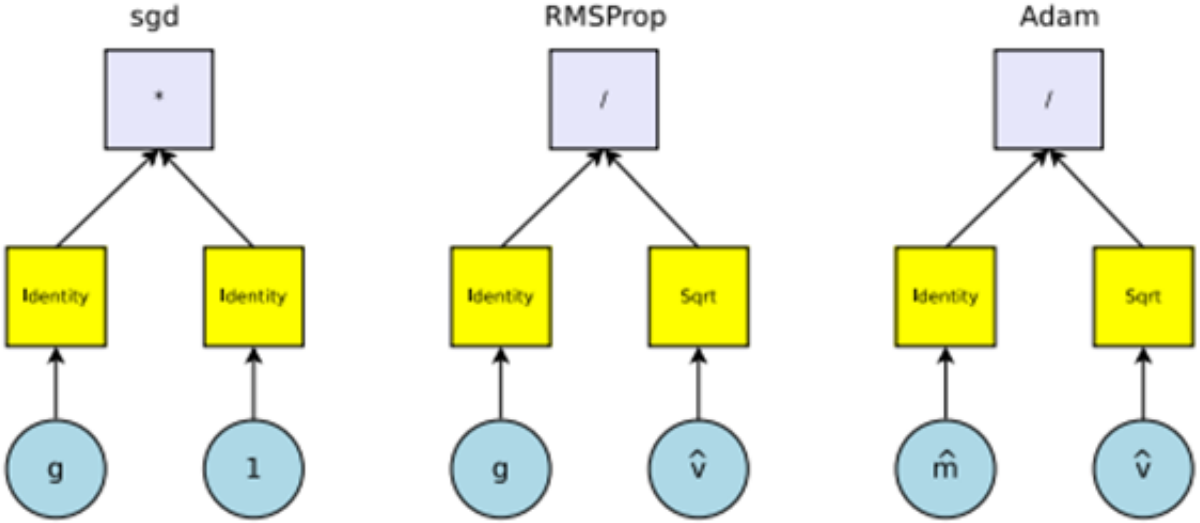


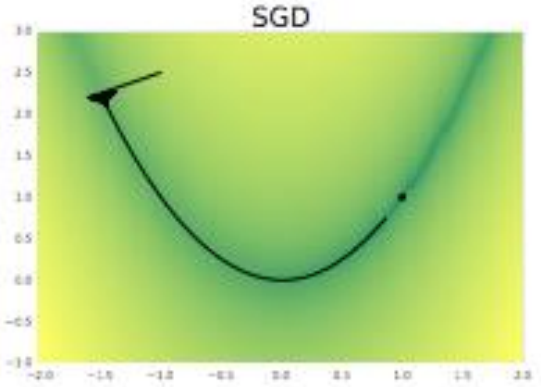
A Full Convolutional Neural Network (LeNet)

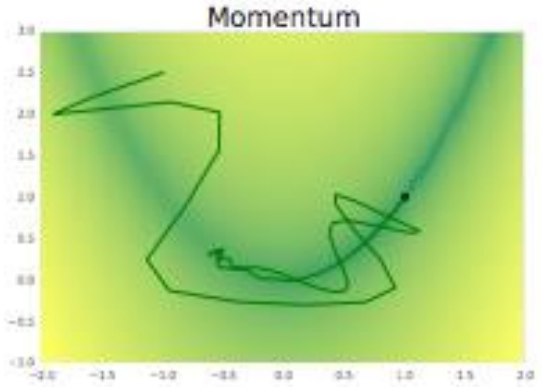
Learning Rate

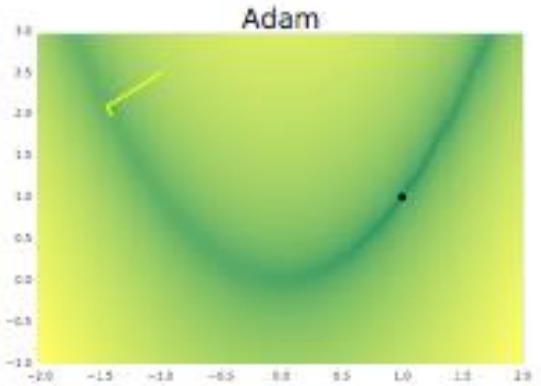


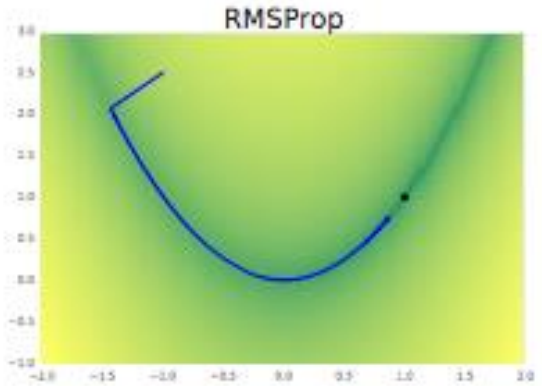
- **Operands**: g, g^2, g^3, $\hat{m}$, $\hat{v}$, $\hat{\gamma}$, $\text{sign}(g)$, $\text{sign}(\hat{m})$, 1, 2, $\epsilon \sim N(0, 0.01)$, $10^{-4}w$, $10^{-3}w$, $10^{-2}w$, $10^{-1}w$, Adam and RMSProp.
- **Unary functions** which map input x to: x, $-x$, e^x, $\log|x|$, $\sqrt{|x|}$, $clip(x, 10^{-5})$, $clip(x, 10^{-4})$, $clip(x, 10^{-3})$, $drop(x, 0.1)$, $drop(x, 0.3)$, $drop(x, 0.5)$ and $\text{sign}(x)$.
- **Binary functions** which map (x, y) to $x + y$ (addition), $x - y$ (subtraction), $x * y$ (multiplication), $\frac{x}{y+\delta}$ (division), x^y (exponentiation) or x (keep left).

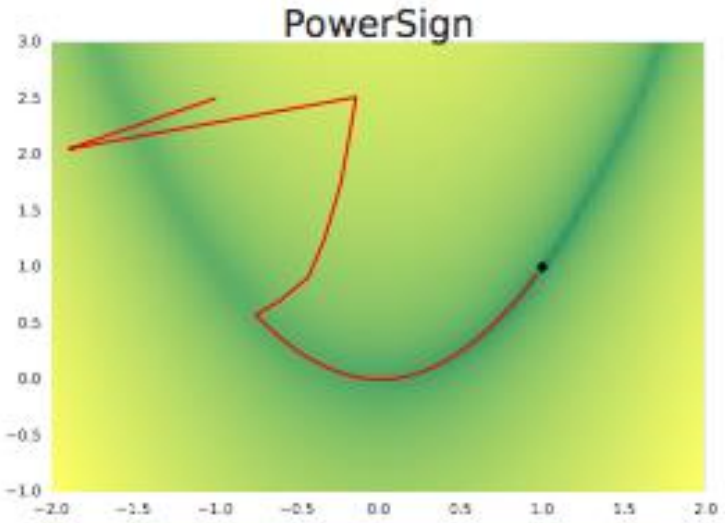








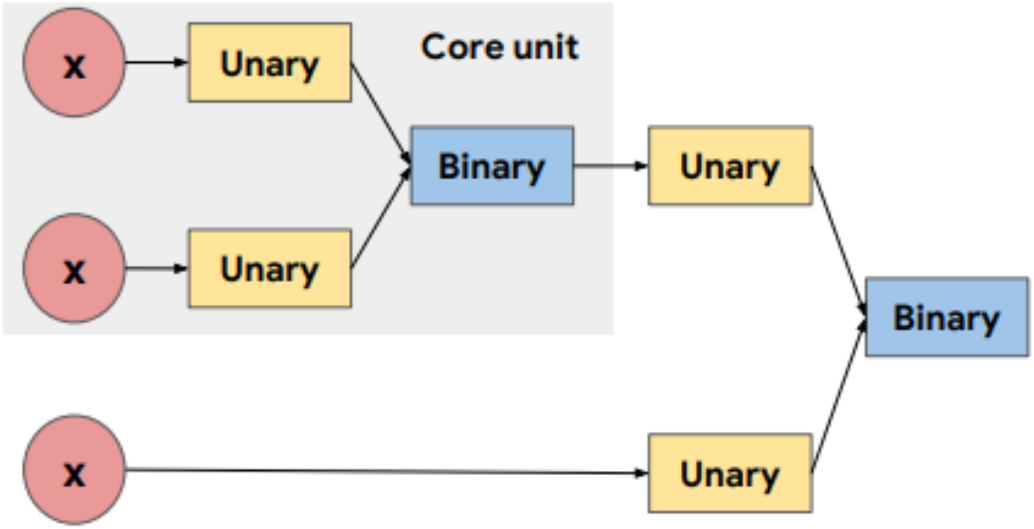




$$e^{\text{sign}(g)*\text{sign}(m)} * g$$

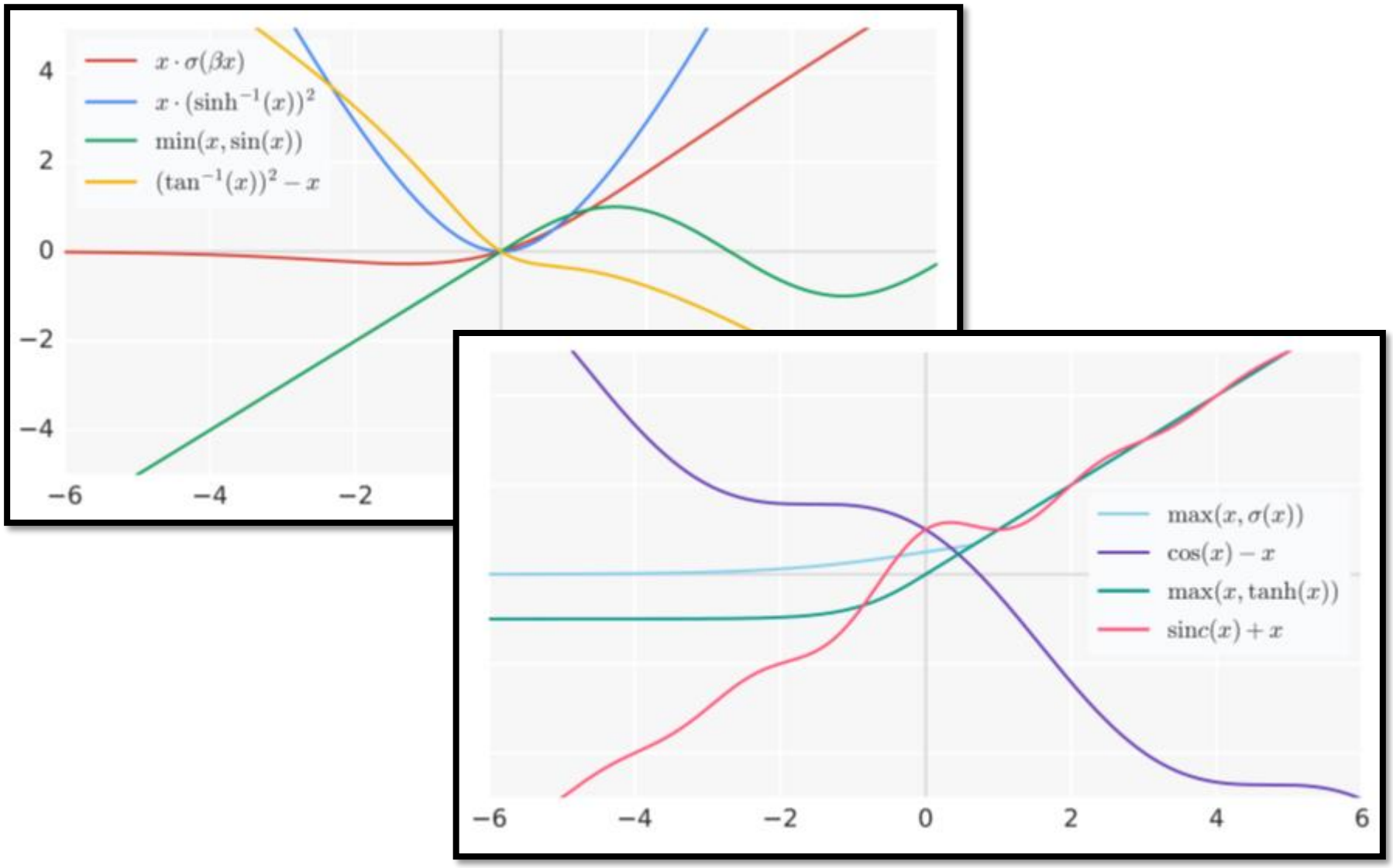
Can transfer to new tasks

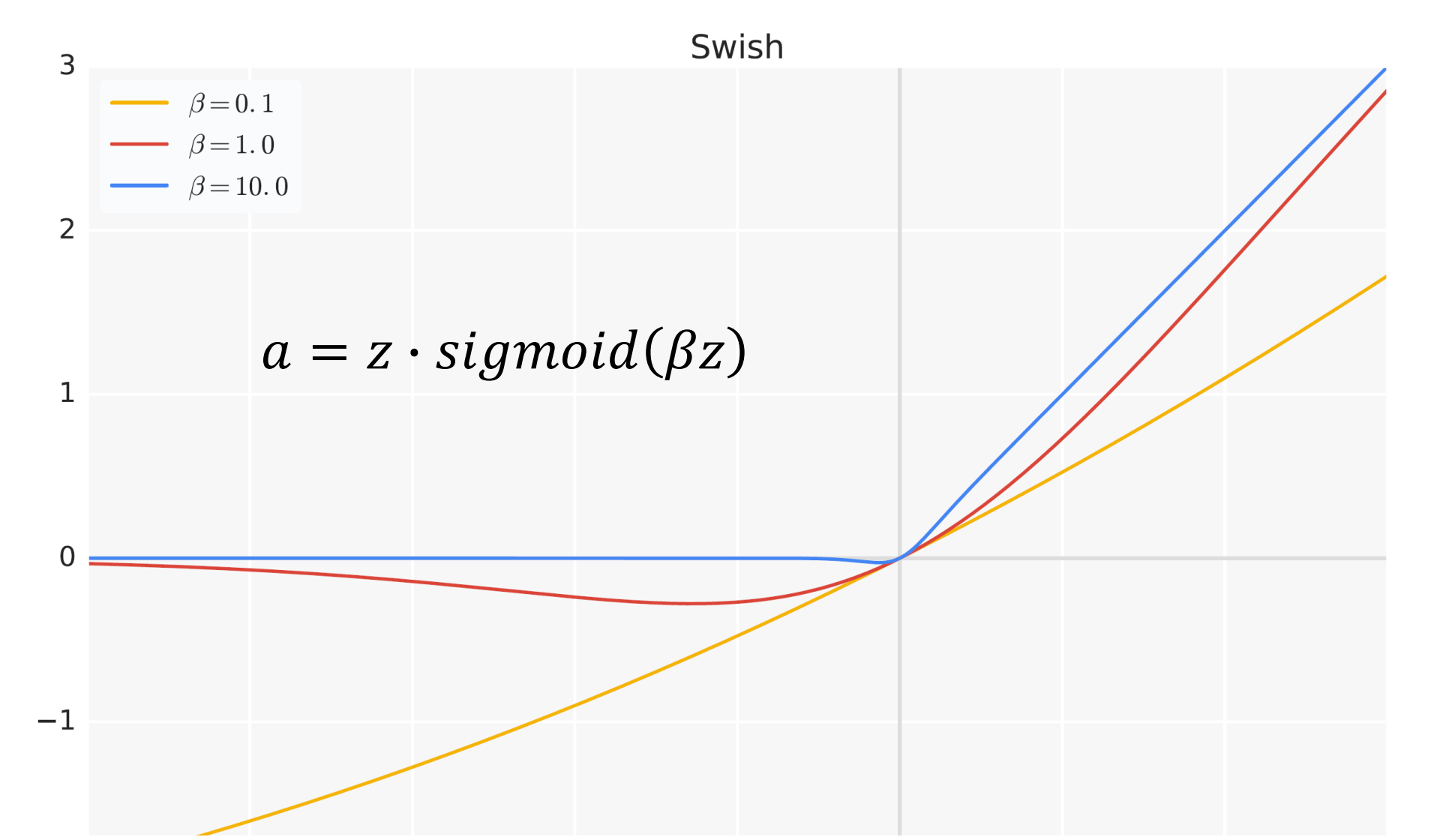
Activation Function



- **Unary functions**: x, $-x$, $|x|$, x^2, x^3, $\sqrt{x}$, βx, $x + \beta$, $\log(|x| + \epsilon)$, $\exp(x)$ $\sin(x)$, $\cos(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\sinh^{-1}(x)$, $\tan^{-1}(x)$, $\text{sinc}(x)$, $\max(x, 0)$, $\min(x, 0)$, $\sigma(x)$, $\log(1 + \exp(x))$, $\exp(-x^2)$, $\text{erf}(x)$, β
- **Binary functions**: $x_1 + x_2$, $x_1 \cdot x_2$, $x_1 - x_2$, $\frac{x_1}{x_2+\epsilon}$, $\max(x_1, x_2)$, $\min(x_1, x_2)$, $\sigma(x_1) \cdot x_2$, $\exp(-\beta(x_1 - x_2)^2)$, $\exp(-\beta|x_1 - x_2|)$, $\beta x_1 + (1 - \beta)x_2$

Activation Function

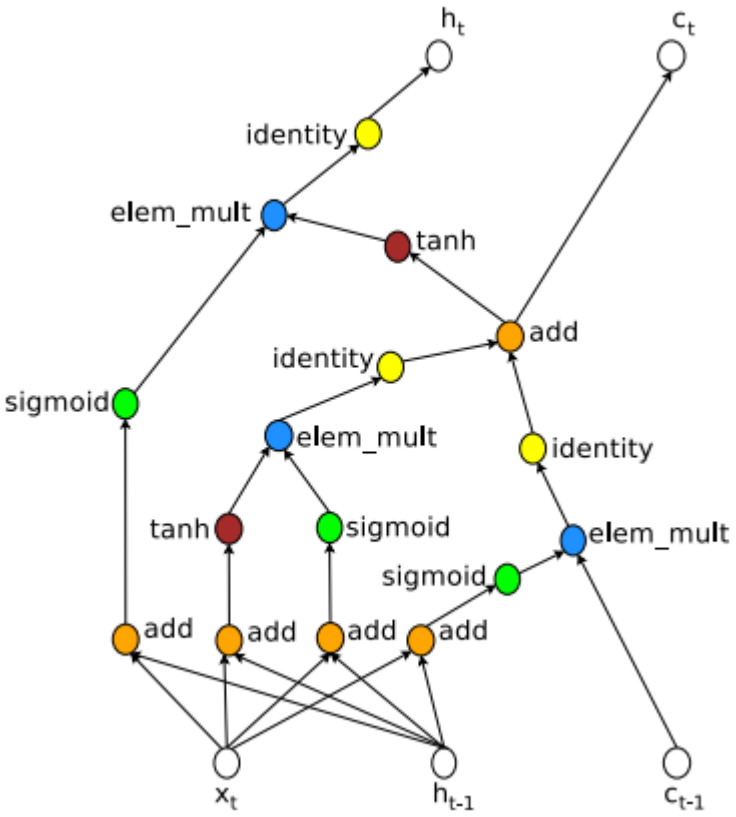




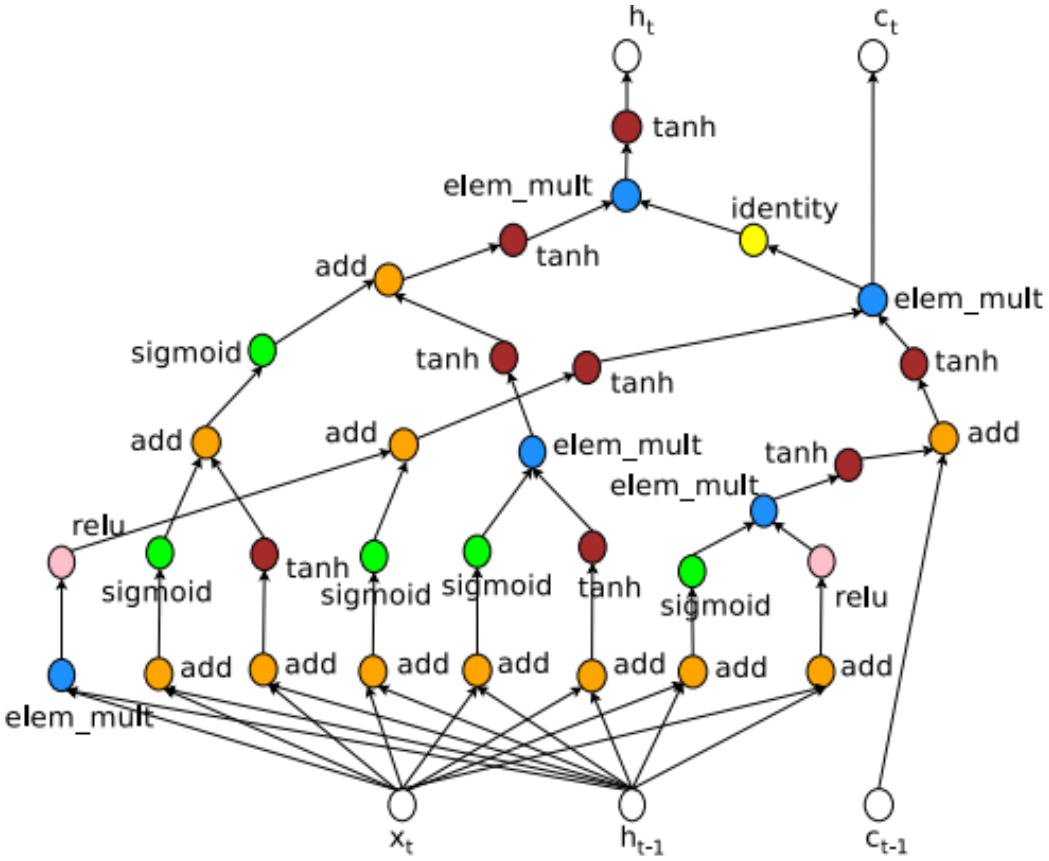
Baselines	**ReLU**	**LReLU**	**PReLU**	**Softplus**	**ELU**	**SELU**	**GELU**
Swish $>$ Baseline	9	7	6	6	8	8	8
Swish $=$ Baseline	0	1	3	2	0	1	1
Swish $<$ Baseline	0	1	0	1	1	0	0

Neural Architecture Search with Reinforcement Learning

LSTM



From Reinforcement Learning



Computation Issue?

- Original version: 450 GPUs for 3-4 days (32,400-43,200 GPU hours).
- New version: Nvidia GTX 1080Ti GPU takes less than 16 hours.
- Main idea: forcing all child models to share weights to instead of training from scratch.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, Jeff Dean, “Efficient Neural Architecture Search via Parameter Sharing”, arXiv, 2018