# Neural Architecture Optimization

Shen Yu 1901111296

2019.7.11

# Motivation

- Substantial effort of human experts on discovering architectures
- Searching the best architecture within discrete space is inefficient

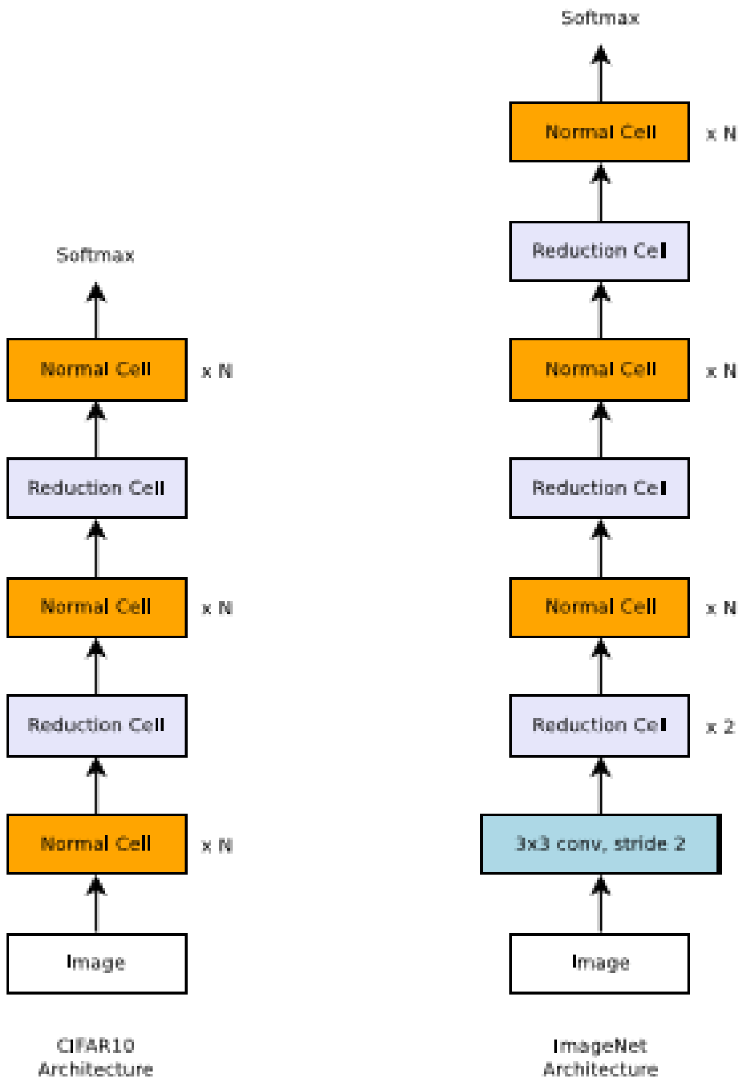
# Contributions

- Propose to optimize network architecture by mapping architectures into a continuous vector space

- Achieve improved efficiency in discovering powerful convolutional and recurrent architectures

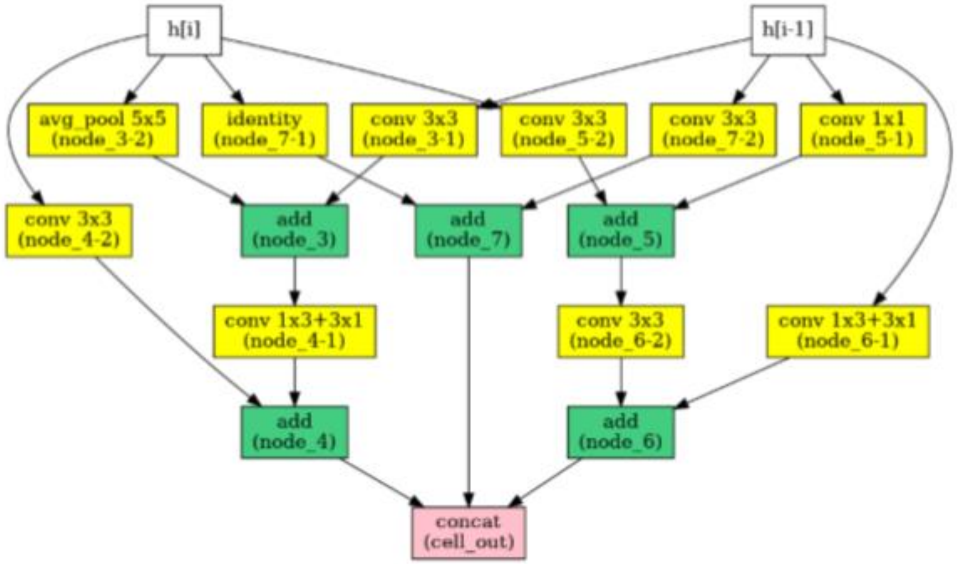
# Related Work

- Reinforcement learning
  - ENAS
- Evolutionary algorithm
  - AmoebaNet
- SMBO
  - PNAS



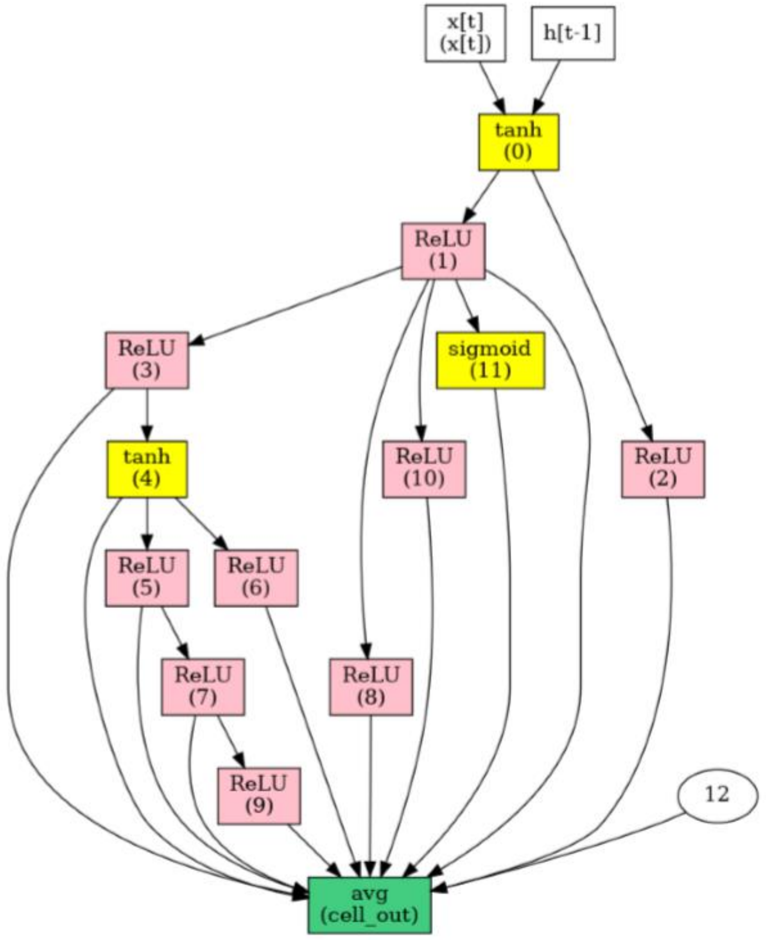
# Architecture space (CNN)

- 2 cells (Normal and Reduction)
  - B(=5) blocks
    - Input 1 from two previous cells or previous blocks
    - Input 2 from two previous cells or previous blocks
    - Operation applied to input 1
    - Operation applied to input 2
  - Output the concatenation of outputs of unused blocks

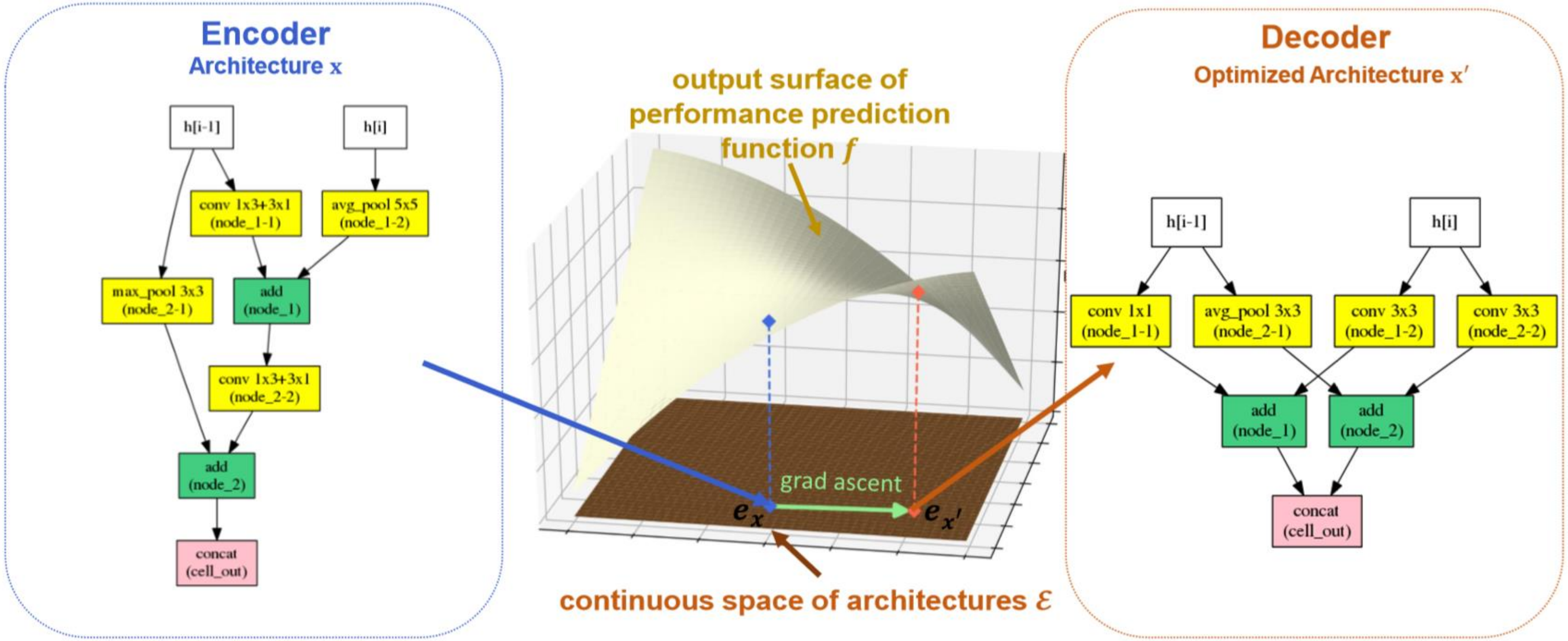


# Architecture space (RNN)

- An RNN cell
  - B(=12) nodes
    - Input from previous nodes
    - Activation
  - Output the average of the outputs of all the nodes



# Overview


Encoder
Architecture x
h[i-1]
h[i]
conv 1x3+3x1 (node_1-1)
avg_pool 5x5 (node_1-2)
max_pool 3x3 (node_2-1)
add (node_1)
conv 1x3+3x1 (node_2-2)
add (node_2)
concat (cell_out)
output surface of performance prediction function $f$
grad ascent
$e_x$
$e_{x'}$
continuous space of architectures $\mathcal{E}$
Decoder
Optimized Architecture x′
h[i-1]
h[i]
conv 1x1 (node_1-1)
avg_pool 3x3 (node_2-1)
conv 3x3 (node_1-2)
conv 3x3 (node_2-2)
add (node_1)
add (node_2)
concat (cell_out)

# Encoder

- Input: A sequence $\{x_1\ x_2\ \ldots\ x_T\}$
  - $x_{example}$ ="node1 conv 3x3 node2 max-pooling 3x3"
- Output: Architecture embedding $e_x$
  - $\{h_1\ h_2\ \ldots\ h_T\} \in R^{T\times d}$, $h_t$ is the hidden state at t-th timestep
- Architecture: A single layer LSTM with $d$ hidden units

# Performance Predictor

- Input: Mean pooling of embedding $e_x$

$$\bar{e}_x = \frac{1}{T}\sum_{t=1}^{T} h_t$$

- Output: Performance prediction

- Architecture: FFNN

# Decoder

- Input: Architecture embedding $e_x$
- Output: Predicted architecture $x'$
- Architecture: LSTM with attention

# Loss function

- Performance predictor

$$L_{pp} = (s_x - f(E(x)))^{2}$$

- Decoder

$$L_{rec} = -\log P(x|E(x)) = -\sum_{t=1}^{T} \log P(x_t|E(x), x_{<t}) = -\sum_{t=1}^{T} \log \frac{\exp(W_{x_t})}{\sum_{x' \in V_t} \exp(W_{x'})}$$

- Final loss

$$L_{final} = \lambda L_{pp} + (1-\lambda) L_{rec}$$

# Algorithm

**Algorithm 1** Neural Architecture Optimization

**Input**: Initial candidate architectures set $X$ to train NAO model. Initial architectures set to be evaluated denoted as $X_{eval} = X$. Performances of architectures $S = \emptyset$. Number of seed architectures $K$. Step size $\eta$. Number of optimization iterations $L$.

**for** $l = 1, \cdots, L$ **do**

Train each architecture $x \in X_{eval}$ and evaluate it to obtain the dev set performances $S_{eval} = \{s_x\}, \forall x \in X_{eval}$. Enlarge $S$: $S = S \bigcup S_{eval}$.

Train encoder $E$, performance predictor $f$ and decoder $D$ by minimizing Eqn.(1), using $X$ and $S$.

Pick $K$ architectures with top $K$ performances among $X$, forming the set of seed architectures $X_{seed}$.

For $x \in X_{seed}$, obtain a better representation $e_{x'}$ from $e_{x'}$ using Eqn. (2), based on encoder $E$ and performance predictor $f$. Denote the set of enhanced representations as $E' = \{e_{x'}\}$.

Decode each $x'$ from $e_{x'}$ using decoder, set $X_{eval}$ as the set of new architectures decoded out: $X_{eval} = \{D(e_{x'}), \forall e_{x'} \in E'\}$. Enlarge $X$ as $X = X \bigcup X_{eval}$.

**end for**

**Output**: The architecture within $X$ with the best performance

# Trick

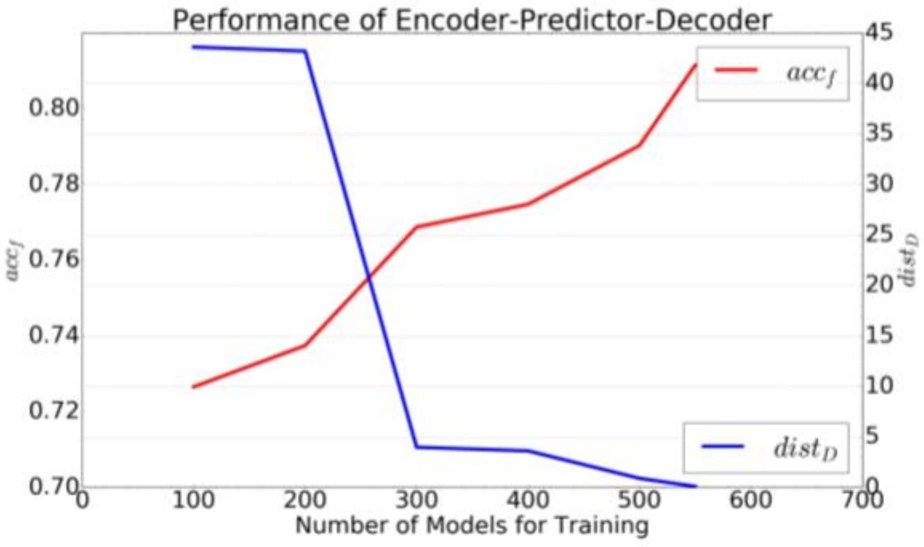
- Data augmentation
  - For each $(x_1, s_x)$, add an additional pair $(x_2, s_x)$ where $x_2$ is symmetrical to $x_1$ and use both pairs to train the encoder and performance predictor

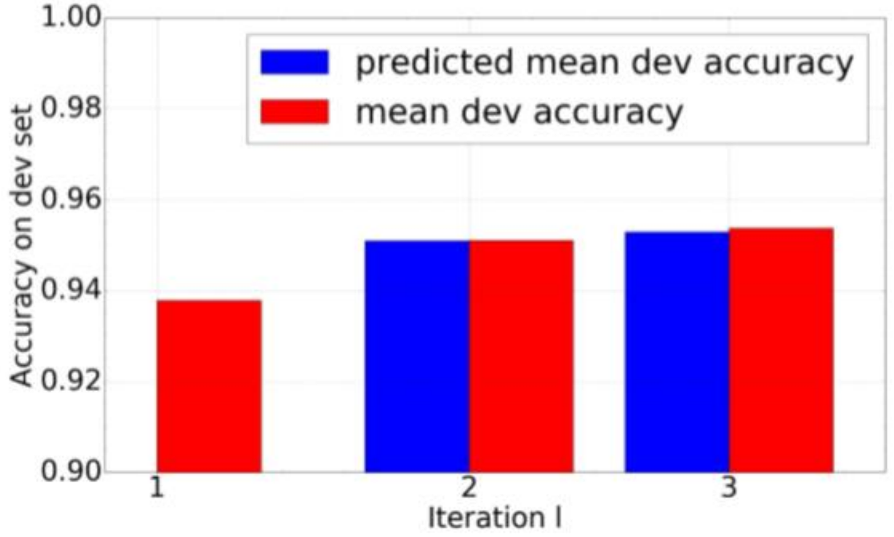
- $x_{example1}$ ="node1 conv 3x3 node2 max-pooling 3x3"
- $x_{example2}$ ="node2 max-pooling 3x3 node1 conv 3x3"
- $s(x_{example1}) = s(x_{example2})$

# Performance on Cifar-10

| Model | B | N | F | #op | Error(%) | #params | M | GPU Days |
|---|---|---|---|---|---|---|---|---|
| DenseNet-BC [19] | | 100 | 40 | / | 3.46 | 25.6M | / | / |
| ResNeXt-29 [43] | | | | / | 3.58 | 68.1M | / | / |
| NASNet-A [47] | 5 | 6 | 32 | 13 | 3.41 | 3.3M | 20000 | 2000 |
| NASNet-B [47] | 5 | 4 | N/A | 13 | 3.73 | 2.6M | 20000 | 2000 |
| NASNet-C [47] | 5 | 4 | N/A | 13 | 3.59 | 3.1M | 20000 | 2000 |
| Hier-EA [27] | 5 | 2 | 64 | 6 | 3.75 | 15.7M | 7000 | 300 |
| AmoebaNet-A [38] | 5 | 6 | 36 | 10 | 3.34 | 3.2M | 20000 | 3150 |
| AmoebaNet-B [38] | 5 | 6 | 36 | 19 | 3.37 | 2.8M | 27000 | 3150 |
| AmoebaNet-B [38] | 5 | 6 | 80 | 19 | 3.04 | 13.7M | 27000 | 3150 |
| AmoebaNet-B [38] | 5 | 6 | 128 | 19 | 2.98 | 34.9M | 27000 | 3150 |
| AmoebaNet-B + Cutout [38] | 5 | 6 | 128 | 19 | 2.13 | 34.9M | 27000 | 3150 |
| PNAS [26] | 5 | 3 | 48 | 8 | 3.41 | 3.2M | 1280 | 225 |
| ENAS [36] | 5 | 5 | 36 | 5 | 3.54 | 4.6M | / | 0.45 |
| Random-WS | 5 | 5 | 36 | 5 | 3.92 | 3.9M | / | 0.25 |
| DARTS + Cutout [28] | 5 | 6 | 36 | 7 | 2.83 | 4.6M | / | 4 |
| NAONet | 5 | 6 | 36 | 11 | 3.18 | 10.6M | 1000 | 200 |
| NAONet | 5 | 6 | 64 | 11 | 2.98 | 28.6M | 1000 | 200 |
| NAONet + Cutout | 5 | 6 | 128 | 11 | 2.11 | 128M | 1000 | 200 |
| NAONet-WS | 5 | 5 | 36 | 5 | 3.53 | 2.5M | / | 0.3 |

# Analysis on NAO



# Performance on Cifar-100

| Model | B | N | F | #op | Error (%) | #params |
|---|---|---|---|---|---|---|
| DenseNet-BC [19] | / | 100 | 40 | / | 17.18 | 25.6M |
| Shake-shake [15] | / | / | / | / | 15.85 | 34.4M |
| Shake-shake + Cutout [11] | / | / | / | / | 15.20 | 34.4M |
| NASNet-A [47] | 5 | 6 | 32 | 13 | 19.70 | 3.3M |
| NASNet-A [47] + Cutout | 5 | 6 | 32 | 13 | 16.58 | 3.3M |
| NASNet-A [47] + Cutout | 5 | 6 | 128 | 13 | 16.03 | 50.9M |
| PNAS [26] | 5 | 3 | 48 | 8 | 19.53 | 3.2M |
| PNAS [26] + Cutout | 5 | 3 | 48 | 8 | 17.63 | 3.2M |
| PNAS [26] + Cutout | 5 | 6 | 128 | 8 | 16.70 | 53.0M |
| ENAS [36] | 5 | 5 | 36 | 5 | 19.43 | 4.6M |
| ENAS [36] + Cutout | 5 | 5 | 36 | 5 | 17.27 | 4.6M |
| ENAS [36] + Cutout | 5 | 5 | 36 | 5 | 16.44 | 52.7M |
| AmoebaNet-B [38] | 5 | 6 | 128 | 19 | 17.66 | 34.9M |
| AmoebaNet-B [38] + Cutout | 5 | 6 | 128 | 19 | 15.80 | 34.9M |
| NAONet + Cutout | 5 | 6 | 36 | 11 | 15.67 | 10.8M |
| NAONet + Cutout | 5 | 6 | 128 | 11 | 14.75 | 128M |

# Performance on PTB

| Models and Techniques | #params | Test Perplexity | GPU Days |
|---|---|---|---|
| Vanilla LSTM [45] | 66M | 78.4 | / |
| LSTM + Zoneout [23] | 66M | 77.4 | / |
| Variational LSTM [14] | 19M | 73.4 | |
| Pointer Sentinel-LSTM [33] | 51M | 70.9 | / |
| Variational LSTM + weight tying [20] | 51M | 68.5 | / |
| Variational Recurrent Highway Network + weight tying [46] | 23M | 65.4 | / |
| 4-layer LSTM + skip connection + averaged weight drop + weight penalty + weight tying [31] | 24M | 58.3 | / |
| LSTM + averaged weight drop + Mixture of Softmax + weight penalty + weight tying [44] | 22M | 56.0 | / |
| NAS + weight tying [47] | 54M | 62.4 | 1e4 CPU days |
| ENAS + weight tying + weight penalty [36] | 24M | 58.6[5] | 0.5 |
| Random-WS + weight tying + weight penalty | 27M | 58.81 | 0.4 |
| DARTS+ weight tying + weight penalty [28] | 23M | 56.1 | 1 |
| NAONet + weight tying + weight penalty | 27M | 56.0 | 300 |
| NAONet-WS + weight tying + weight penalty | 27M | 56.6 | 0.4 |

# Q&A