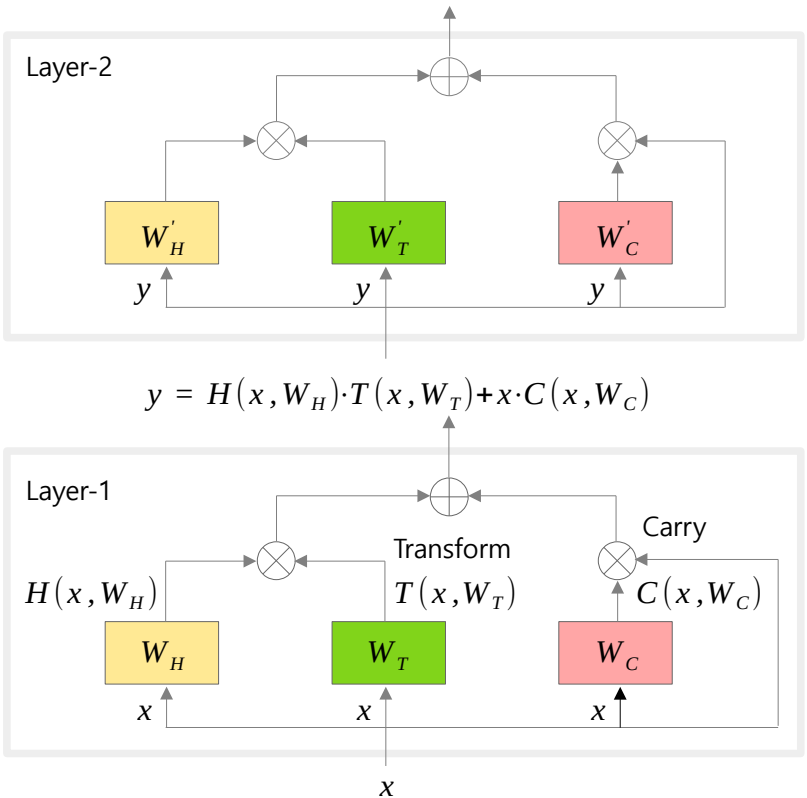


9. Highway Networks

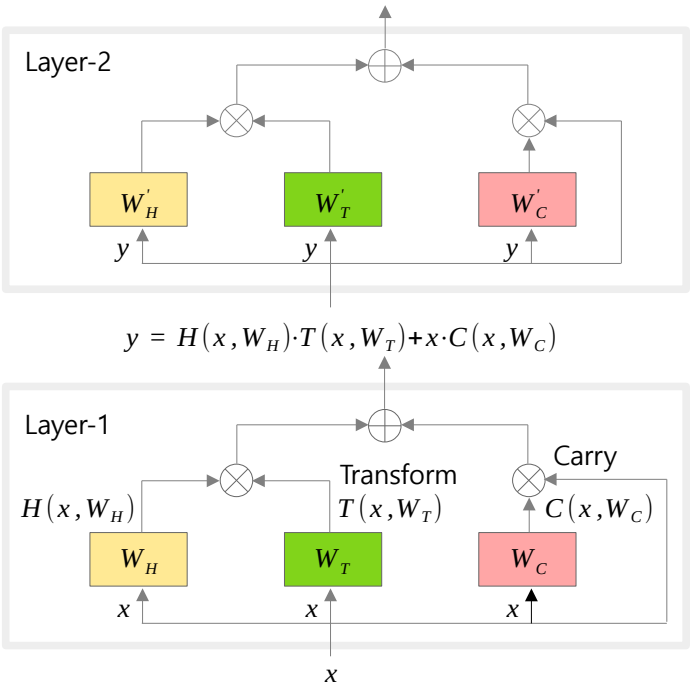
This video was produced in Korean and translated into English, and the audio was generated by AI (TTS).

www.youtube.com/@meanxai

MX-AI

Highway Networks

1. Paper
2. Highway Network Structure
3. Performance
4. Implementing a Highway Network
5. Residual Connections



MX-AI

■ Highway Network paper (2015)

Highway Networks

Rupesh Kumar Srivastava RUPESH@IDSIA.CH
Klaus Greff KLAUS@IDSIA.CH
Jurgen Schmidhuber JUERGEN@IDSIA.CH

The Swiss AI Lab IDSIA
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale
Universita della Svizzera italiana (USI)
Scuola universitaria professionale della Svizzera italiana (SUPSI)
Galleria 2, 6928 Manno-Lugano, Switzerland

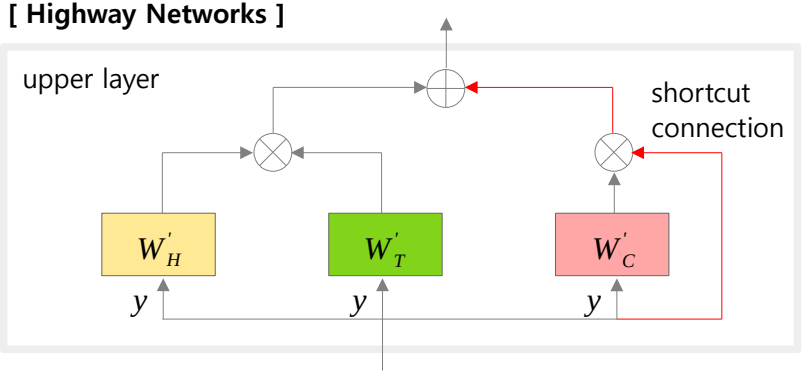
Abstract

There is plenty of theoretical and empirical evidence that depth of neural networks is a crucial ingredient for their success. However, network training becomes more difficult with increasing depth and training of very deep networks remains an open problem. In this extended abstract, we introduce a new architecture designed to ease gradient-based training of **very** deep networks. We refer to networks with this architecture as highway networks, since they allow unimpeded information flow across several layers on information highways. The architecture is characterized by the use of gating units which learn to regulate the flow of information through a network. Highway networks with hundreds of layers can be trained directly using stochastic gradient descent and with a variety of activation functions, opening up the possibility of studying extremely deep and efficient architectures.

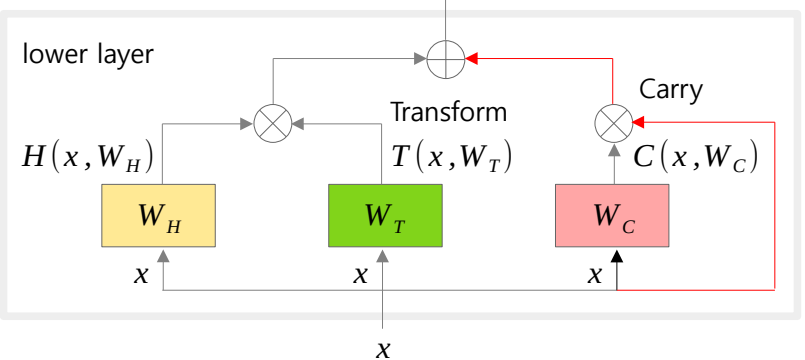
Highway Network: Structure

- The authors of the paper proposed highway networks that allow unimpeded information flow across multiple layers of the information highways. During forward propagation, the input signals flow to the upper layers via the highways. Additionally, during backpropagation, gradients flow through the highways to the lower layers.

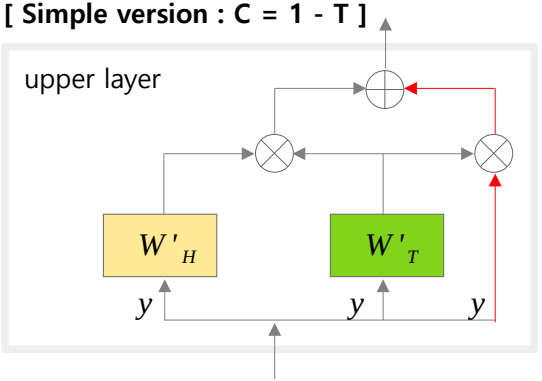
[Highway Networks]



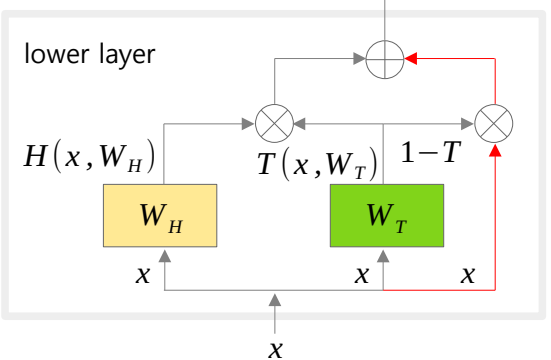
$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$



[Simple version : C = 1 - T]



Eq. 3: $y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$



$$y = \begin{cases} x, & \text{if } T(x, W_T) = 0 \\ H(x, W_H) & \text{if } T(x, W_T) = 1 \end{cases}$$

W_H and W_T are determined through learning. If T is 0, 100% of the input signal x is passed on to the upper layer. and if T is 1, the shortcut connection is closed and only H is passed to the upper layer, just like the existing feedforward network. This is the T-weighted average of x and H. This is similar to how cell states propagate to the next stage in LSTM.

T is a kind of weight, and it uses the sigmoid activation function to make the output be between 0 and 1.

The dimensionality of x, y, $H(x,W_H)$ and $T(x,W_T)$ must be the same for Equation (3) to be valid.

MX-AI

■ Highway Network: Performance

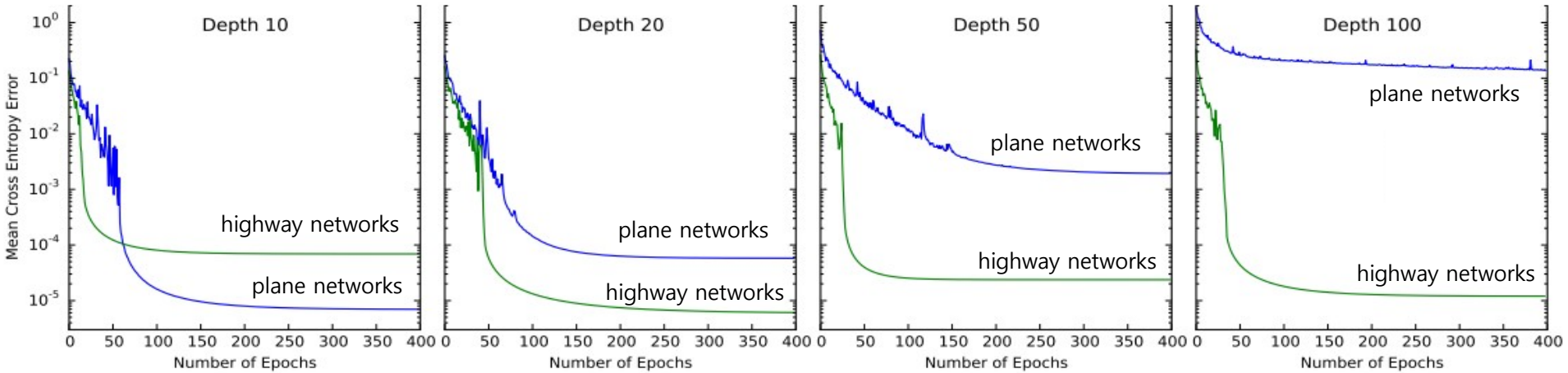


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.

* When the depth is about 10, the performance of plain networks is better, but from the depth of 20, the performance of highway networks gets better as the depth increases.

MX-AI

■ Highway Network: Performance

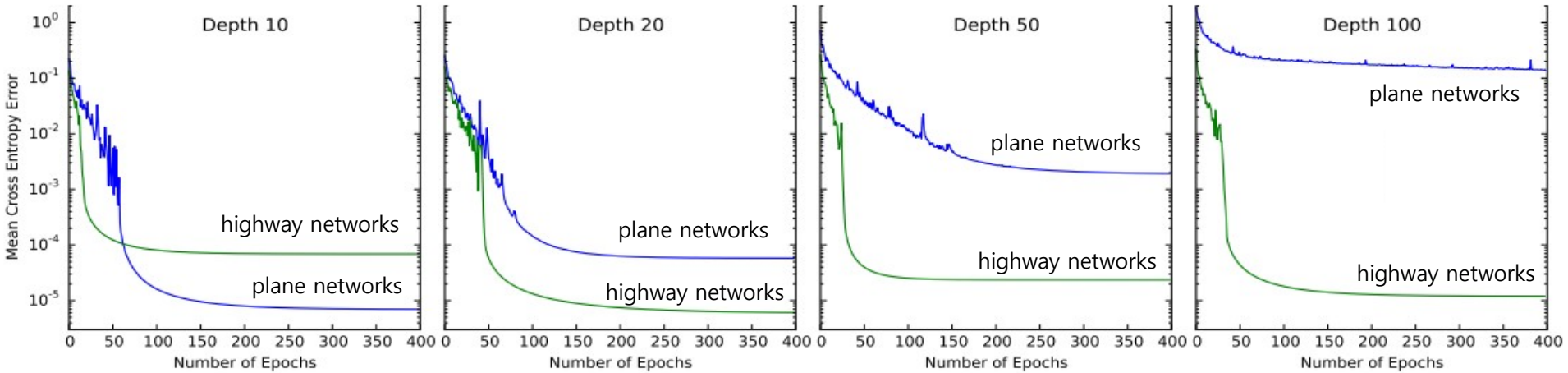


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.

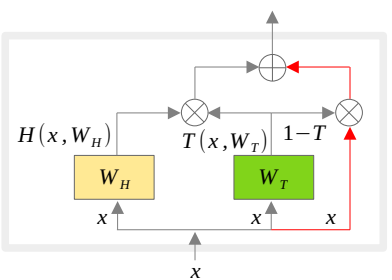
* When the depth is about 10, the performance of plain networks is better, but from the depth of 20, the performance of highway networks gets better as the depth increases.

MX-AI

■ Implementing the Highway Networks

```
# [MXDL-9-01] 1.highway_network.py
import numpy as np
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Model
from sklearn.model_selection import train_test_split
from sklearn.datasets import fetch_openml
import matplotlib.pyplot as plt
import pickle

# Read MNIST dataset
# x, y = fetch_openml('mnist_784', return_X_y=True)
# x = np.array(x) / 255
# y = np.array(y.to_numpy().astype('int8')).reshape(-1,1)
# with open('mnist.pkl', 'wb') as f:
#     pickle.dump([x, y], f, pickle.HIGHEST_PROTOCOL)
with open('mnist.pkl', 'rb') as f:
      x, y = pickle.load(f)

x_train,x_test,y_train,y_test=train_test_split(x, y)
n_class = len(set(y_train.reshape(-1,)))

# Highway Networks
# y = H(x, WH).T(x, WT) + x(1 - T(x, WT))
def HighWay(x, n_layers):

    for i in range(n_layers):
        H = Dense(x.shape[-1],
                  kernel_initializer='he_normal',
                  bias_initializer='zeros',
                  activation = 'relu')(x)
```

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot (1 - T(x, W_T))$$



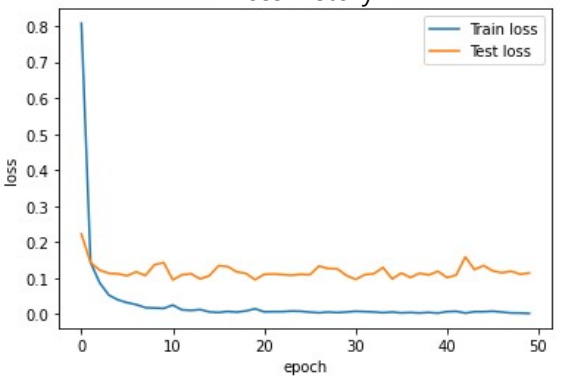

```
        T = Dense(x.shape[-1],
                  kernel_initializer='glorot_normal',
                  bias_initializer = 'zeros',
                  activation = 'sigmoid')(x)

        x = H * T + x * (1. - T)
    return x

# Create an ANN model with Highway networks
x_input = Input(batch_shape=(None, x_train.shape[1]))
h = HighWay(x_input, n_layers = 20)
y_output = Dense(n_class, activation='softmax')(h)

model = Model(x_input, y_output)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer='adam')
model.summary()

# Training
hist = model.fit(x_train, y_train,
                 batch_size=1000,
                 epochs=50,
                 validation_data = (x_test, y_test))

# Visually see the loss history
plt.plot(hist.history['loss'], label='Train loss')
plt.plot(hist.history['val_loss'], label='Test loss')
plt.legend()
plt.title("Loss history")
plt.xlabel("epoch")
plt.ylabel("loss")
plt.show()
```

MX-AI

■ Implementing the Highway Networks

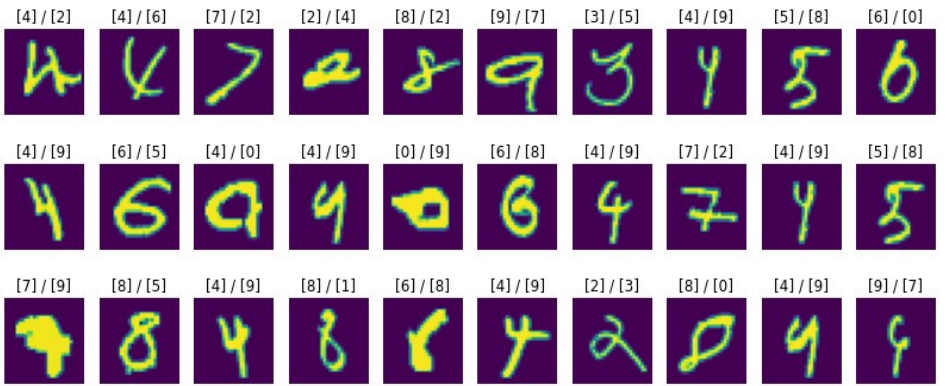
```
y_prob = model.predict(x_test)
y_pred = np.argmax(y_prob, axis=1).reshape(-1,1)
acc = (y_test == y_pred).mean()
print('Accuracy of the test data ={:.4f}'.format(acc))

# Let's check out some misclassified images.
n_sample = 10
miss_cls = np.where(y_test != y_pred)[0]
miss_sam = np.random.choice(miss_cls, n_sample)

fig, ax = plt.subplots(1, n_sample, figsize=(14,4))
for i, miss in enumerate(miss_sam):
    x = x_test[miss] * 255
    ax[i].imshow(x.reshape(28, 28))
    ax[i].axis('off')
    ax[i].set_title(str(y_test[miss]) + ' / ' + str(y_pred[miss]))


Epoch 1/50
53/53 [=========] - 6s 93ms/step - loss: 0.8074 - val_loss: 0.2227
Epoch 2/50
53/53 [=========] - 5s 88ms/step - loss: 0.1436 - val_loss: 0.1434
Epoch 3/50
53/53 [=========] - 5s 88ms/step - loss: 0.0870 - val_loss: 0.1224
…

Epoch 49/50
53/53 [=========] - 5s 86ms/step - loss: 0.0033 - val_loss: 0.1112
Epoch 50/50
53/53 [=========] - 5s 86ms/step - loss: 0.0023 - val_loss: 0.1144
```

Loss history



Accuracy of the test data =0.9827

* Examples of misclassified image:



MX-AI

■ Residual Network paper (2015)

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. ...

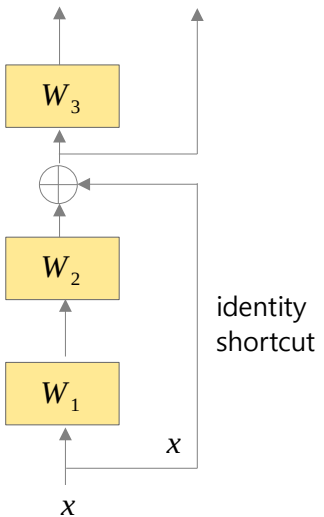
2. Related Work

Shortcut Connections.

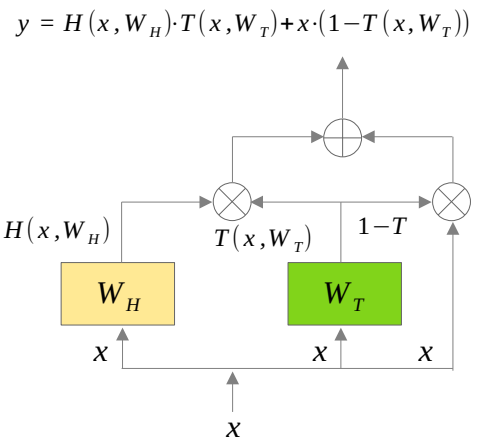
...

Concurrent with our work, "highway networks" present shortcut connections with gating functions. These gates are data-dependent and have parameters, in contrast to our identity shortcuts that are parameter-free. When a gated shortcut is "closed" (approaching zero), the layers in highway networks represent non-residual functions. On the contrary, our formulation always learns residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned.

[Residual Network]



[Simple Highway Network]



MX-AI

■ Residual Network

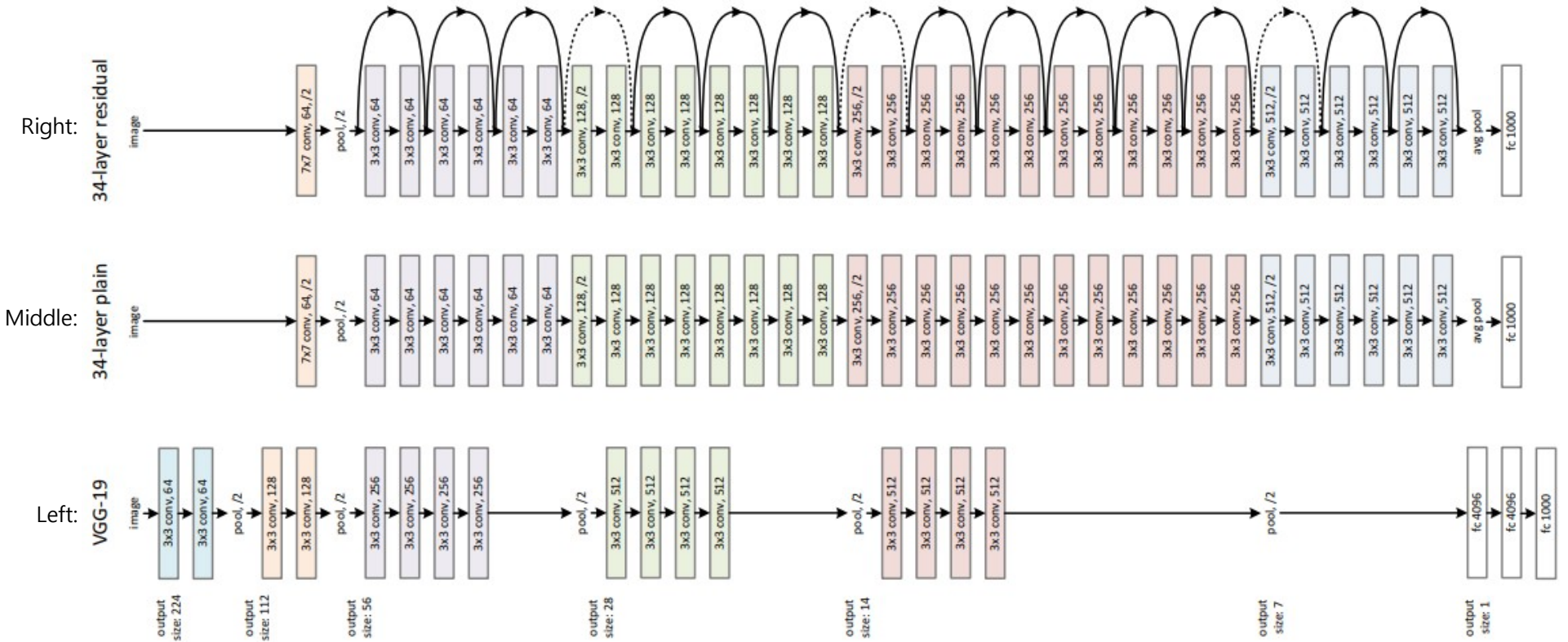


Figure 3. Example network architectures for ImageNet. Left: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. Middle: a plain network with 34 parameter layers (3.6 billion FLOPs). Right: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. Table 1 shows more details and other variants.

MX-AI

- Transformer paper (2017)

Attention Is All You Need

Ashish Vaswani∗ Google Brain avaswani@google.com
Noam Shazeer∗ Google Brain noam@google.com
Niki Parmar∗ Google Research nikip@google.com
Jakob Uszkoreit∗ Google Research usz@google.com

Llion Jones∗ Google Research llion@google.com
Aidan N. Gomez∗ University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser∗ Google Brain lukaszkaiser@google.com

Illia Polosukhin∗ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 Englishto-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data

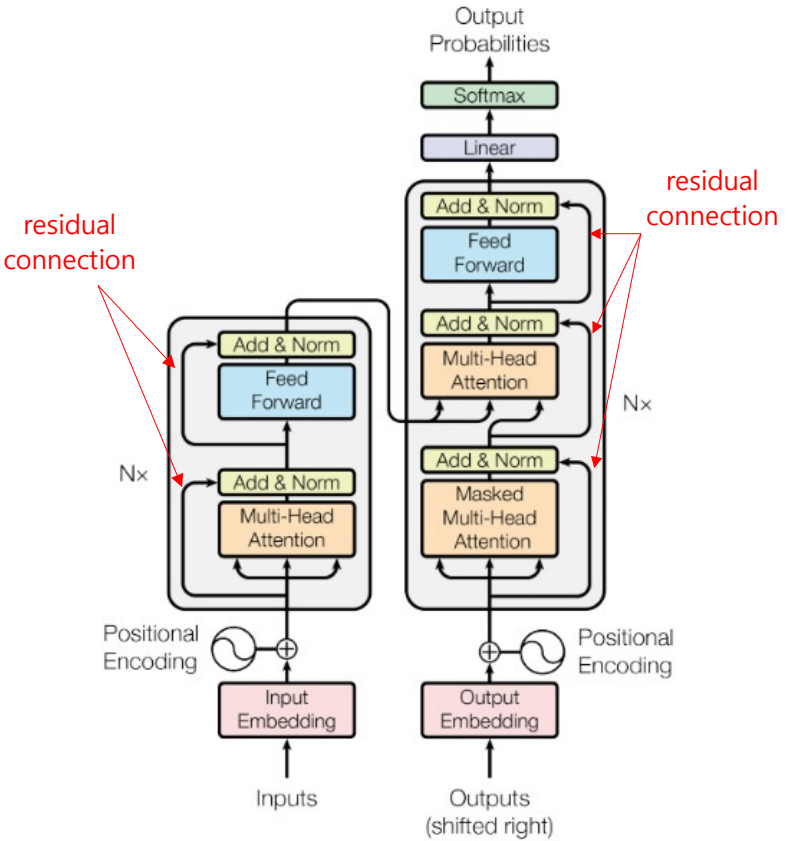


Figure 1: The Transformer - model architecture.