

# Word2Vec with TensorFlow

Department of Computer Science,  
National Tsing Hua University, Taiwan

Deep Learning

# Word embedding

- How to represent words in machine learning?

*My favorite pet is cat.*

- Treat words as discrete atomic symbols, and therefore ‘cat’ may be represented as ‘1’ and ‘pet’ as ‘2.’
- However, this embedding includes no semantics. Tasks learning from this representation causes models hard to train.
- Better solutions?

# Word embedding

- Intuition: Modeling relationships between words.
  - Semantics Similarity
    - ◎ CBOW (Continuous Bag-of-Words model),
    - ◎ Skip-Gram
  - Grammar / Syntax

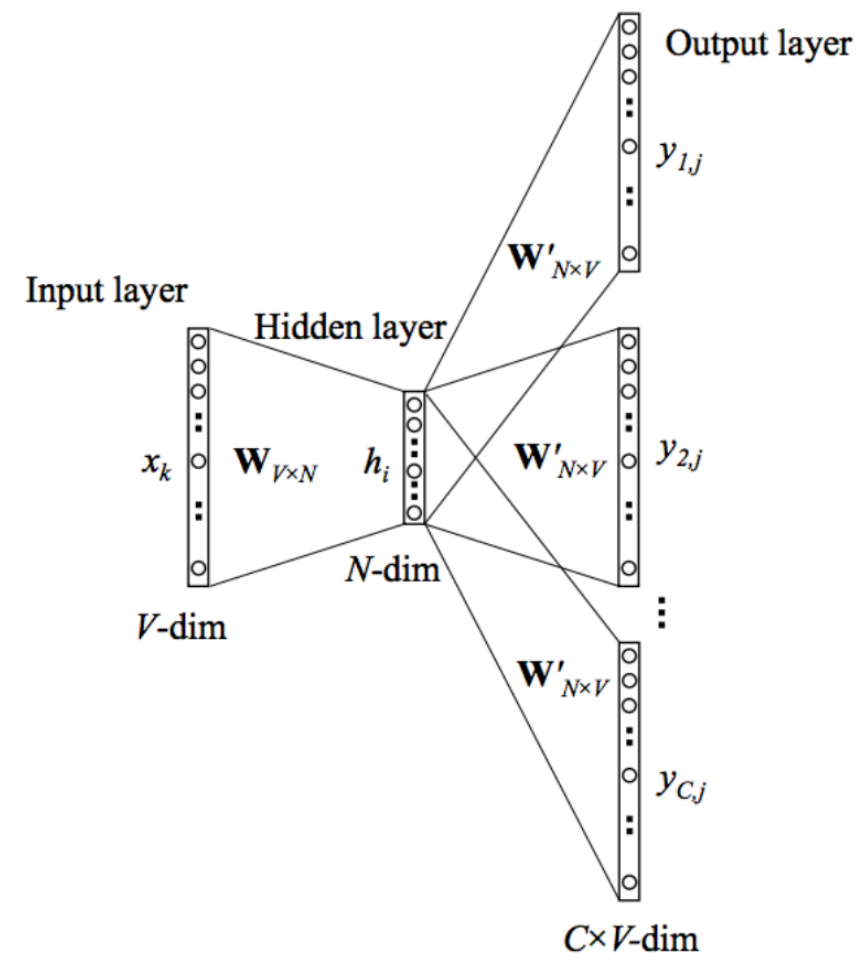
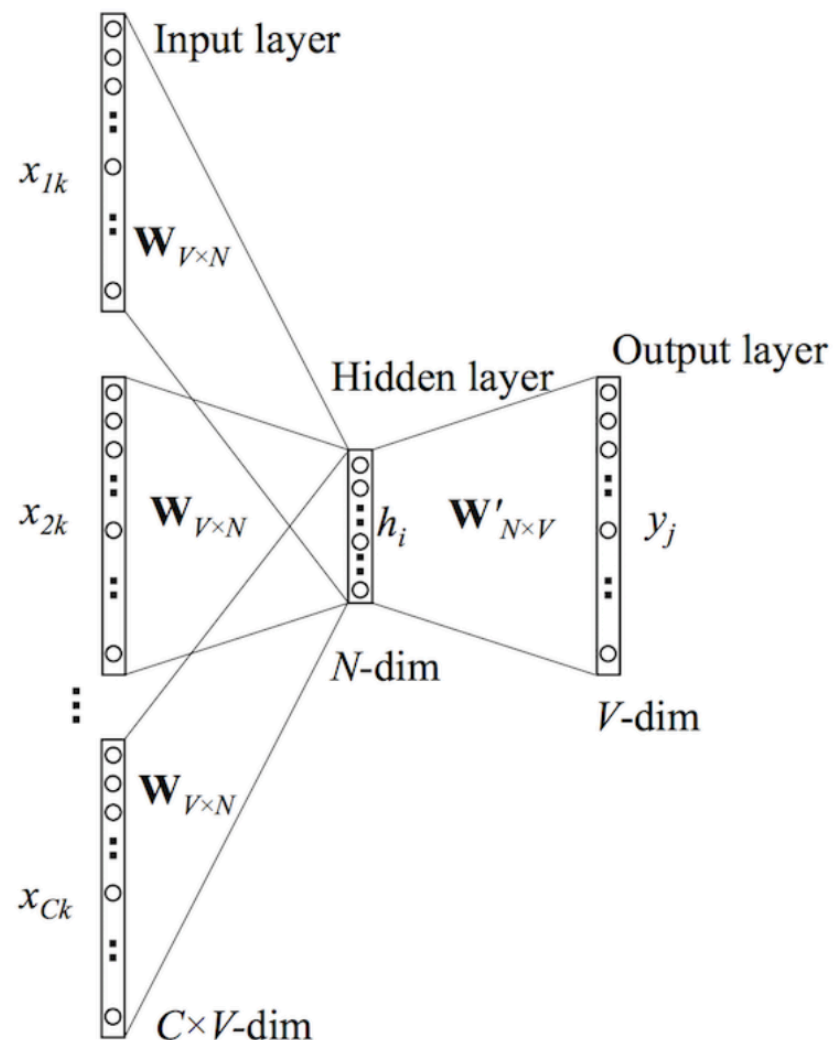
# CBOW & Skip-Gram

- Assumptions in CBOW & Skip-Gram:
  - Words with similar meaning / domain have the close contexts.

*My favorite pet is \_\_\_\_ . cat, dog*

# CBOW & Skip-Gram

- CBOW: Predicting the target words with context words.
- Skip-Gram: Predicting context words from the target words.



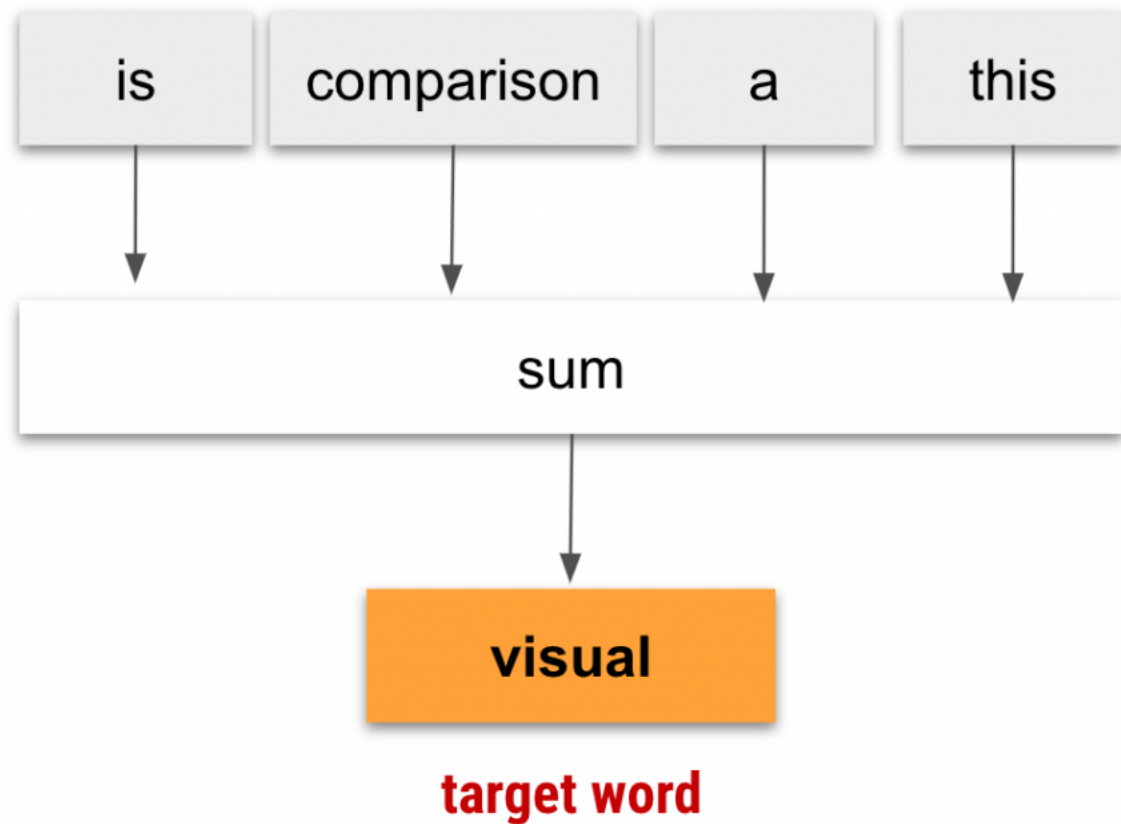
# CBOW & Skip-Gram

- For example, given the sentence:

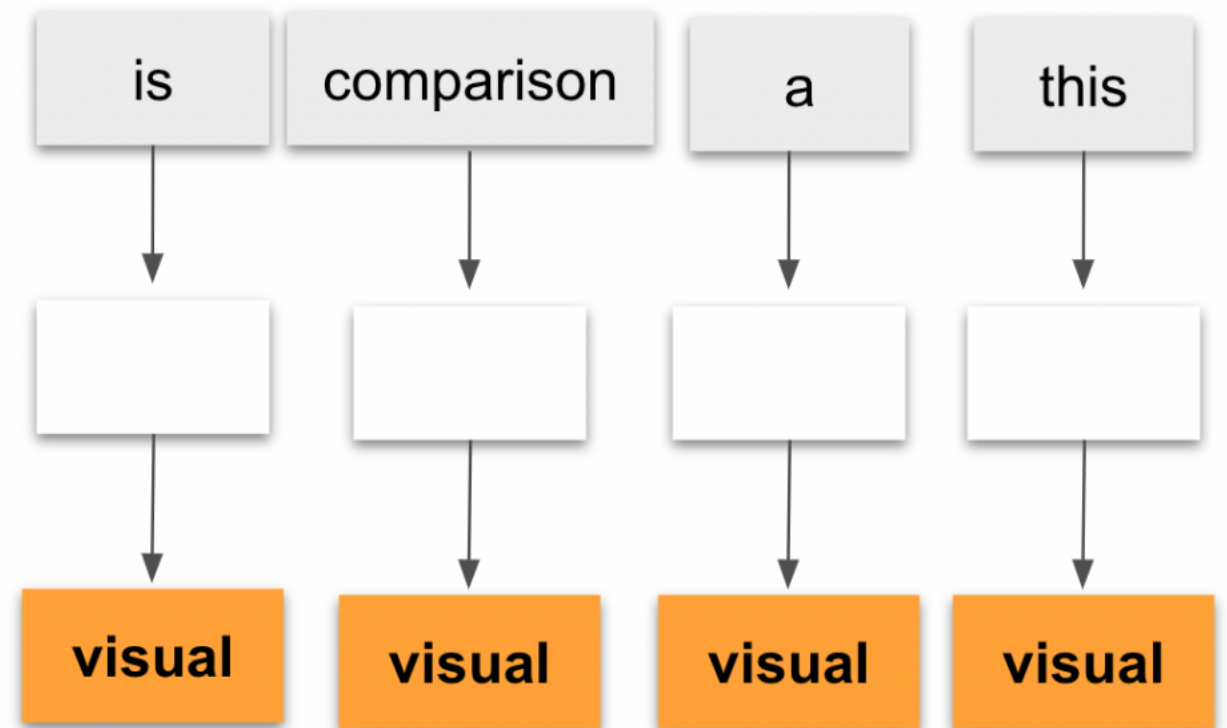
**the quick brown fox jumped over the lazy dog.**

- CBOW will be trained on the dataset:  
([the, brown], quick), ([quick, fox], brown), ...
- Skip-Gram will be trained on the dataset:  
(quick, the), (quick, brown), (brown, quick),  
(brown, fox)...

## CBOW



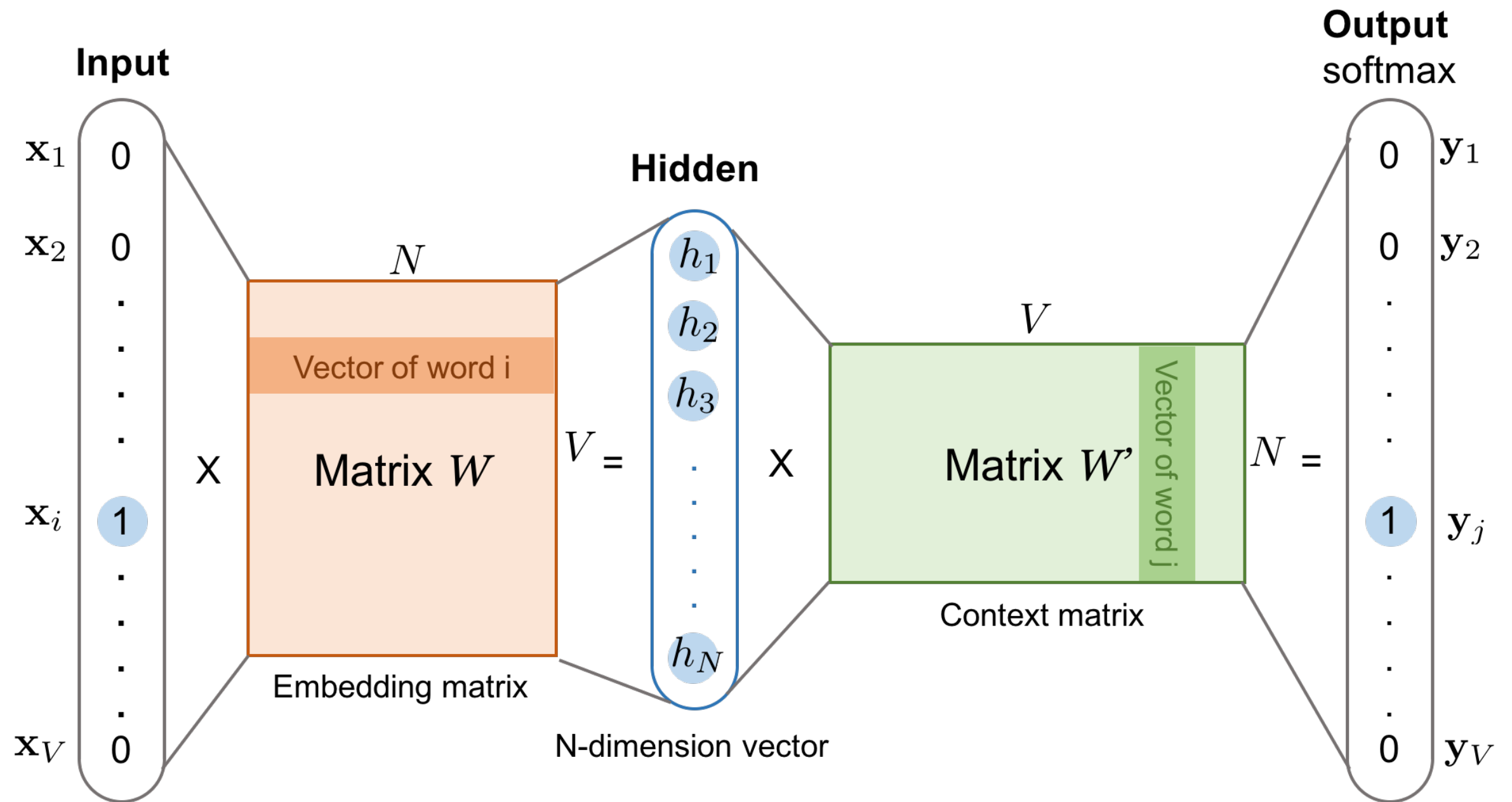
## SkipGram



By: Kavita Ganesan

This is a visual comparison

# Skip-Gram





# Softmax & Words classification

- Recap the softmax function:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

- While training the embedding, we need to sum over the entire vocabulary in the denominator, which is costly.

$$p(w = i | \mathbf{x}) = \frac{\exp(h^\top v_{w'_i})}{\sum_{j \in V} \exp(h^\top v_{w'_j})}$$

**| V | (vocabulary size) is thousands of millions**

# Noise Contrastive Estimation

- Intuition: Estimate the parameters by learning to discriminate between the data  $w$  and some artificially generated noise  $\tilde{w}$ .

*maximum likelihood estimation  $\rightarrow$  noise contrastive estimation*

- Objective function:

$$\arg \max \log p(d = 1 | w_t, x) + \sum_{\tilde{w}_i \sim Q}^N \log p(d = 0 | \tilde{w}_i, x)$$

- Derive form:

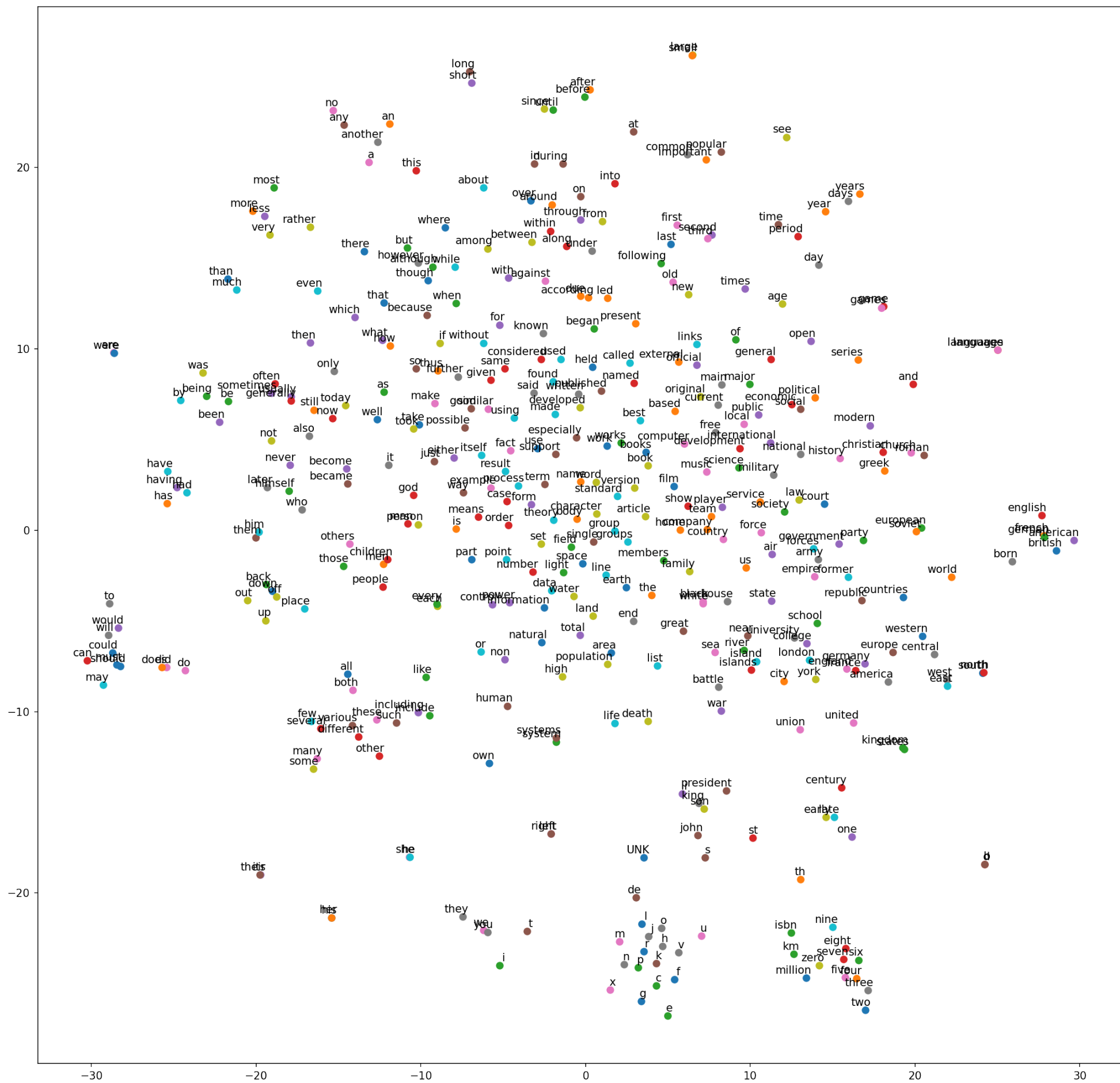
$$\arg \max \log \frac{p(w_t | x)}{p(w_t | x) + Nq(\tilde{w})} + \sum_{\tilde{w}_i \sim Q}^N \log \frac{Nq(\tilde{w}_i)}{p(w_t | x) + Nq(\tilde{w}_i)}$$

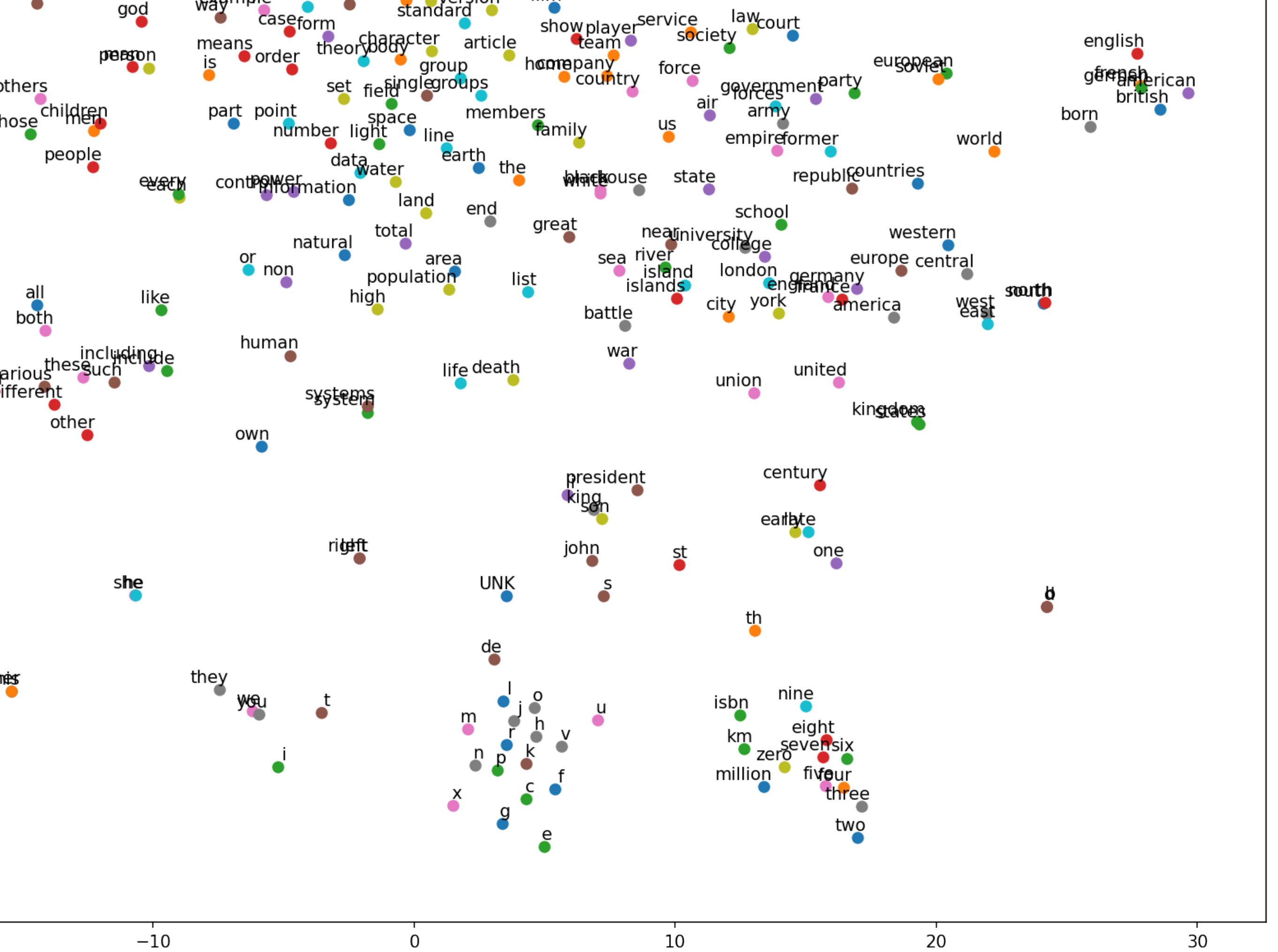
# Noise Contrastive Estimation

- Intuition: Estimate the parameters by learning to discriminate between the data  $w$  and some artificially generated noise  $\tilde{w}$ .


*maximum likelihood estimation  $\rightarrow$  noise contrastive estimation*

- After training: evaluate with softmax function.





# The Neural History of Natural Language Processing

- 
- 2001 ● Neural language models
  - 2008 ● Multi-task learning
  - 2013 ● Word embeddings
  - 2013 ● Neural networks for NLP
  - 2014 ● Sequence-to-sequence models
  - 2015 ● Attention
  - 2015 ● Memory-based networks
  - 2018 ● Pretrained language models

2018 ● Pretrained language models

# Homework

- Goal: practice model subclassing
  - Train your word2vec via **model subclassing**
  - Deadline: 2020-11-5(Thur) 23:59

# Homework

- Functional API:

```
from tensorflow.keras import Model, Input

center_words = Input(shape=(), name='center_words', dtype='int32')
target_words = Input(shape=(1), name='target_words', dtype='int32')

embedding = embedding_lookup()(center_words)
loss = nce_loss()(embedding, target_words)

word2vec = Model(name='word2vec',
                  inputs=[center_words, target_words],
                  outputs=[loss])
```



# Homework

- Model subclassing:

```
class ResNet(tf.keras.Model):  
  
    def __init__(self):  
        super(ResNet, self).__init__()  
        self.block_1 = ResNetBlock()  
        self.block_2 = ResNetBlock()  
        self.global_pool = layers.GlobalAveragePooling2D()  
        self.classifier = Dense(num_classes)  
  
    def call(self, inputs):  
        x = self.block_1(inputs)  
        x = self.block_2(x)  
        x = self.global_pool(x)  
        return self.classifier(x)  
  
resnet = ResNet()
```

# Reference

- <https://ruder.io/word-embeddings-softmax/index.html#noisecontrastiveestimation>
- <https://lilianweng.github.io/lil-log/2017/10/15/learning-word-embedding.html>
- [https://aegis4048.github.io/optimize\\_computational\\_efficiency\\_of\\_skip-gram\\_with\\_negative\\_sampling](https://aegis4048.github.io/optimize_computational_efficiency_of_skip-gram_with_negative_sampling)
- <http://proceedings.mlr.press/v9/gutmann10a/gutmann10a.pdf>
- <https://papers.nips.cc/paper/5165-learning-word-embeddings-efficiently-with-noise-contrastive-estimation.pdf>
- [https://www.tensorflow.org/api\\_docs/python/tf/nn/nce\\_loss](https://www.tensorflow.org/api_docs/python/tf/nn/nce_loss)