Position-wise Feed-Forward Networks

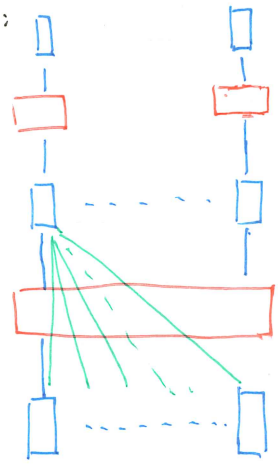
- Applied to each position separately and identically

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

W_1: project to $d = 2048$ 输入扩大四倍(中间隐藏层)

W_2: project back to $d = 512$

Transformer:



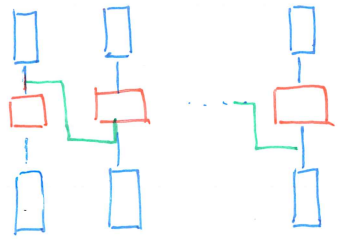
MLP 增强信息活化能力，在抽象层面表示更多，映射成我更想要的语义空间

single MLP 即可，历史信息已被汇聚

(全局)

抓取序列中信息，并做汇聚 (aggregation)

RN



时序信息来自上一时刻

No.

Date

Embedding

We use learned embeddings to convert the input tokens and output tokens to vectors of d_{model} (512)

· Same weight matrix between two embedding layers and the pre-softmax linear transformation.

· multiply weights by $\sqrt{d_{model}}$ 在学embedding时L2long会比较小，乘以d以匹配Positional Encoding的scale（维度越大的向量归一化后单个值越小）

Positional Encoding.

Attention中无时序信息

$PE(pos,2i) = \sin(pos/10000^{2i/d})$

$PE(pos,2i+1) = \cos(pos/10000^{2i/d})$

用周期不同的sin和cos函数计算用来表示词的向量（d=512）

与Input embedding直接相加，两个都在[-1,1]附近

Layer Type.	Complexity / Layer	Sequential Operation (计算便利度(多少能够平行计算))	Max Path length. (learn long-range dependency) (信息整合性)
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention. (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Campus Attention缺点：需要更多数据和更大模型，贵，跑小训练集容易过拟合

Training

25000 source and target tokens.

NVIDIA P100 : 8

Base model: 0.4 sec / training step

100,000 steps

big models: 1.0 sec / training step

300,000 steps

Adam optimizer with $\beta_1 = 0.9$ $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$.

$$\text{learning rate} = d_{\text{model}}^{-0.5} \cdot \min(\text{step\_num}^{-0.5}, \text{step\_num} \cdot \text{warmup\_steps}^{-1.5})$$

4000

模型越宽向量越长，学习率越低

从小到大

爬到最高后按 0.5 次方衰减

Regularization:

① Residual Dropout. 在子层(Multi-head, MLP)的输出进入残差连接和进入 layer-alone 前做 dropout. $P_{drop} = 0.1$ → 10% 的×0，剩下 9% 乘以 $\frac{1.0}{0.9}$

在 embedding 与 positional encoding 之后也 dropout 0.1

② Label smoothing: 对于正确的词 softmax 只需到 0.9（实际很难逼近 1）

置信度

但会增加模型不确信度

No.

Date

Improving Language Understanding by Generative Pre-Training

No.
Date

generative
GPT：生在未标号（unlabeled）文件上生成一个预训练模型，再在有标号的子任务上训练分辨率的微调模型

在微调时构造任务相关（task-aware）的输入很少改变架构

Pre-trained word embedding can improve performance in many NLP tasks
词嵌入模型

困难：①. 不知道用什么优化目标函数.（optimization objectives）
各有所长

②. 难以把学到的表示传递到下游子任务上面
子任务差异大，没有统一有效的方式

Semi-supervised —— Combinations of unsupervised pretraining and supervised finetuning

架构：Transformer：

相比于RNN，transformer 在迁移学习时学到的特征更稳健（robust）

原因：Transformer 具有更结构化的记忆（structured memory）
能处理更长文本，抽取句子和段落层面的语义信息.

- 使用任务相关的输入表示

Unsupervised Pre-training

☆ To maximize the following likelihood

$$L_1(u) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \theta)$$ 此处求和是因为log放回去就是∏

(hp) k: context window.

使用 Transformer 的解码器（只有decoder 才有掩码）

The model applies a multi-headed self-attention operation over the input context tokens followed by position-wise feed forward layers

$U = (U_{-k} \dots U_{-1})$ is the context vector of tokens,

n is the number of layers.

We: Token Embedding Matrix Wp: Position Embedding Matrix

$h_0 = UW_e + W_p$

$h_l = \text{transformer\_block}(h_{l-1}) \quad \forall i \in [1, n]$

$P(u) = \text{softmax}(h_n W_e^T)$

BERT: 做完形，用 Transformer 的编码器

GPT: 用更难的目标函数，预测未来，解码器，更高天花板

Supervised fine-tuning

dataset C: each instance consists of a sequence of input tokens $x^1, x^2 \dots x^m$ along with a label y

W_y: Linear output layer to predict y (parameter)

$P(y \mid x^1, \dots x^m) = \text{softmax}(h_l^m W_y)$

☆ To maximize the following objective.

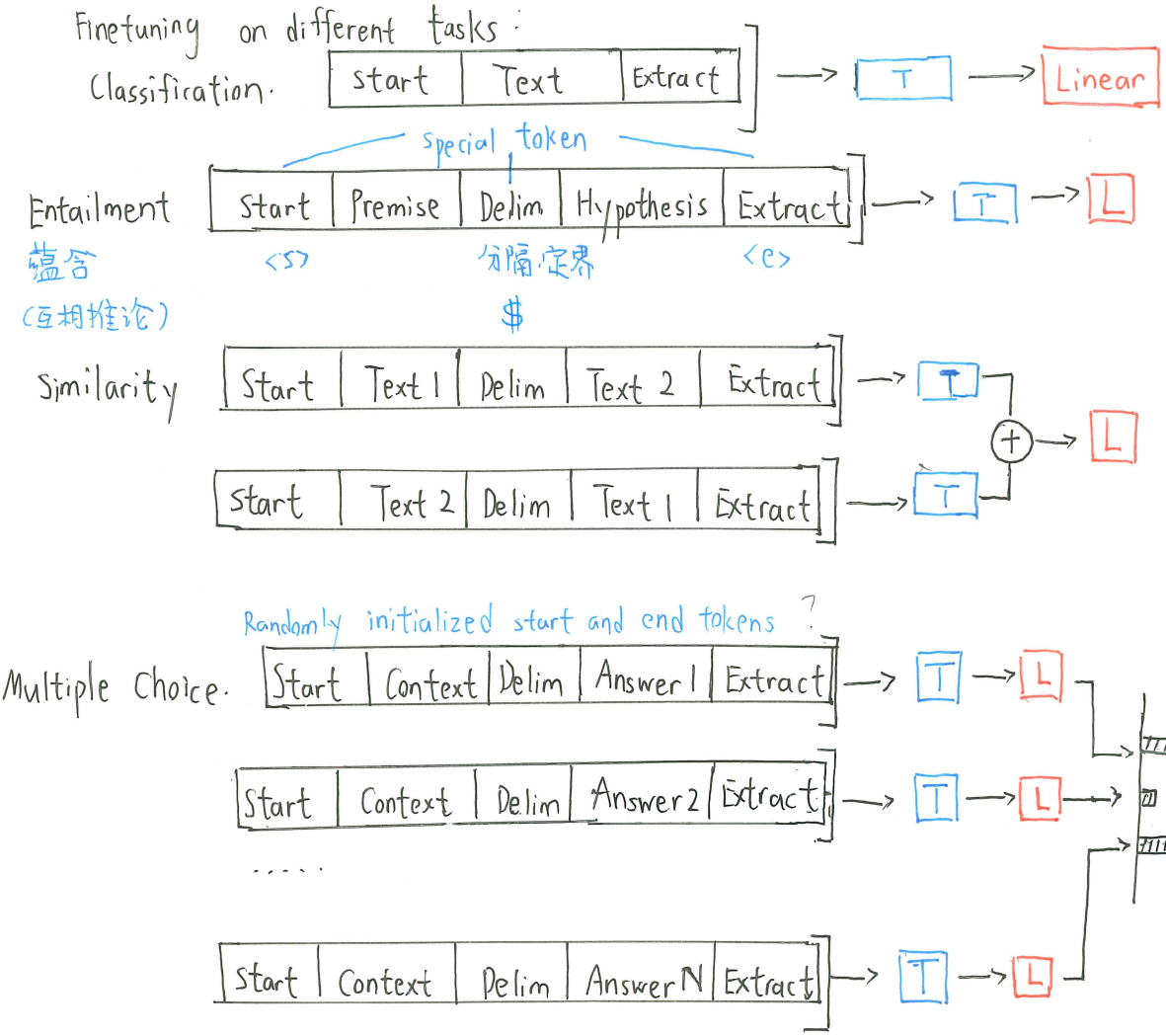
$L_2(C) = \sum_{(x,y)} \log P(y \mid x^1, \dots x^m)$

预测序列的标号

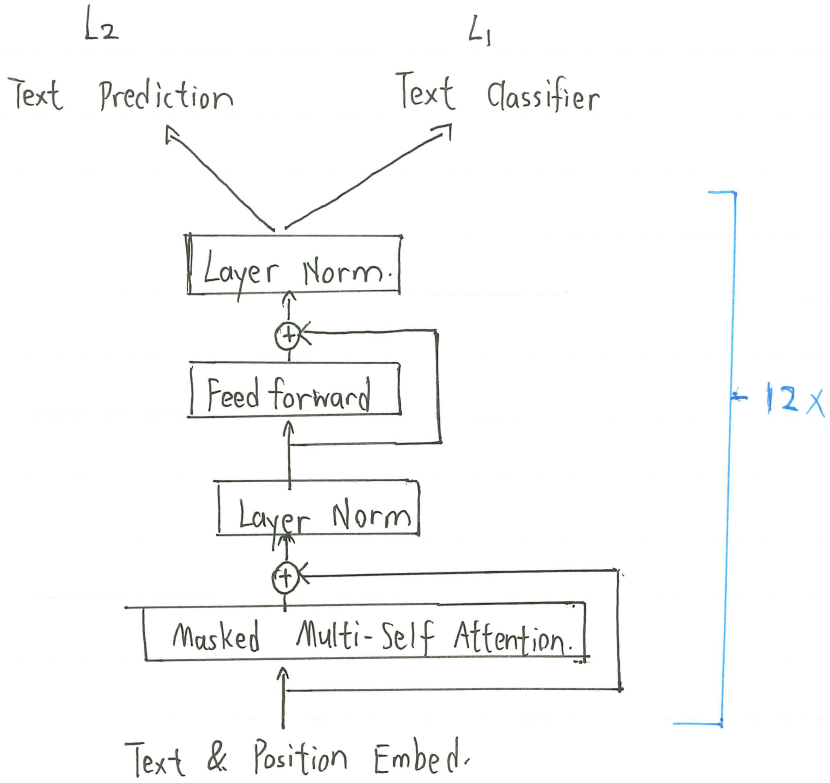
$$L_3(C) = L_2(C) + \lambda * L_1(C)$$

预测下一个词　超参

Finetuning on different tasks:

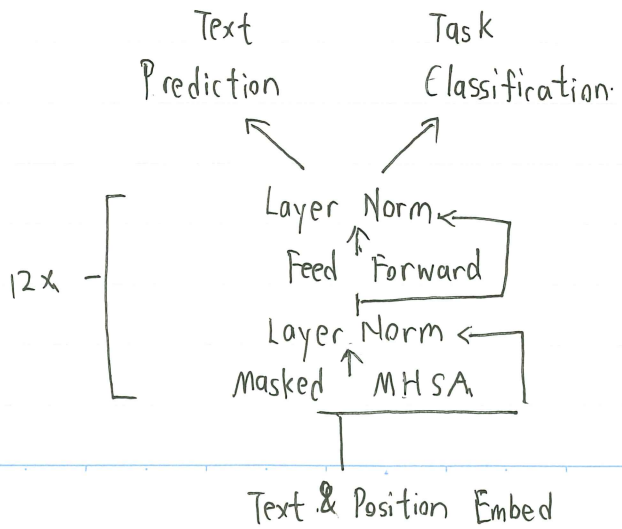


输入形式和输出构造无论怎么变，Transformer 模型不变



BERT的数据集是GPT的4倍，精度更高

GPT-2：



Language Models are Unsupervised Multitask Learners

GPT2. 数据集(wed text)

从新的角度看问题：Zero-shot，更好泛化性

Estimate conditional distribution p(output|input, task)

Language provides a flexible way to specify tasks, inputs and outputs all as a sequence of symbols.

e.g. (answer the question, document, question, answer)

whether objective is supervised doesn't influence its global minimum.

爬了reddit上 kamma>3 的数据

Difference between GPT-2 Architecture. different from the transformer model. such as.

	GPT-2	Transformer.
Decoder Layers	12	6
Attention Heads	12	8
Dimensional	768	512
Context size (for positional encoding	1024	5000.
Layer	GeLU	ReLU
Capture subwords	BytePair Encoding	Tokenization

Language Models are Few-Shot Learners

GPT 3 : 回到 Few-shot，通过大量实验。

175 billion parameters, not sparse (contain a lot of 0)

Applied without any gradient updates or fine-tuning.

微调效果好不代表大模型泛化好

* in-context learning — 在 multi task 中各任务内的上下文

无梯度更新；通过注意力机制处理长序列

Zero shot

Translate English to French: task description

cheese => prompt

One shot: task description

sea otter => butre de mer ⟶ example

prompt.

缺点：难以载入长的新训练样本

每次都要往回全部抓取。

GPT 2,3 相比于 GPT 1 的改变：① modified initialization

② : pre-normalization

③ reversible tokenization 反转词元

* 计算复杂度与宽度是平方关系，和层数是线性关系

使用相对来说大的批量：计算性能↑，并行度↑，通讯量↓。

对于小模型降低批量大小可以增加噪音

模型变大时过拟合没有那么严重，同时学习率向下减

No.

Date

训练数据基于过滤后的Common Crawl ①基于GPT2数据分类过滤

② LSH算法去重（data retrieval

③ 增加高质量数据集

训练过程：分布式训练，模型分割，数据分割

局限性：文本生成新意度弱，没有推进，适合补全

2、语言生成没有重点（模型会花很多时间学虚词）。能否分类以优化？

3. 欠缺多模态

4. 样本有效性不够

5、贵

6. 决策方式未知（权重代表什么）

疑问：是基于样本学习还是在数据库中找相关？

No.

Date

Principle of prompt engineering

Principle 1. Write clear and specific instructions

- To avoid prompt injections

Tactic1: Use delimiters.

triple quotes: """

triple backticks: ```

triple dashes: ---

Angle brackets: < >

XML tags: <tag> </tag>

2. Tactic 2: Ask for structured output

Tactic 3: check whether conditions are satisfied

whether there're prompts?

Tactic 4. Few-shot prompting

Principle 2: Given the model time to think.

Tactic 1: specify the steps to complete the task

Tactic 2.: Instruct the model to work out its own solution before rushing to a conclusion

otherwise the model might neglect mistakes in the process

Reducing hallucinations:

First find relevant information, then

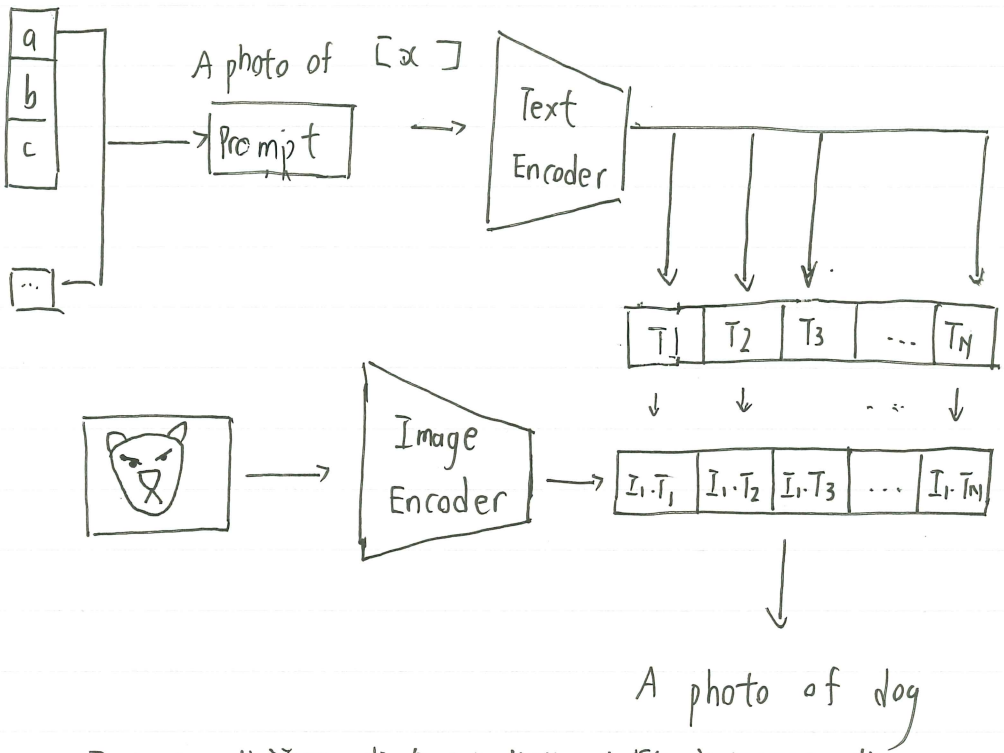
Answer the question based on evidence.

Prompt engineering is an iterative process

No.

Date

CLIP



- 通过对比式学习，省略生成式学习中繁琐实例细节，在抽象语义级别特征空间区分数据，拥有更简单的优化，更好的泛化
- 自回归预测 / 掩码完形填空 ——> 自监督训练
  目标函数与下游任务无关，而想要得到泛化能力强的特征

方法不新，但数据规模和模型规模实无前例

Transformer 的 Deep contextual representation learning 赋上下文语义环境学习方式，带来自监督范式下取之不尽用之不竭的文本监督信号

0.4 billion 数据（训练样本）

CV 与 NLP 的结合 使得能够学习多模态的特征，
很容易做 zero-shot 的迁移学习

把 训练任务变成对比任务 —> 效率 x4
（学习）

- Prompt engineering 的意义：
  ① 多义性
  ② 预训练时基本都是句子，只输入单词，抽取出的特征可能不好
- CLIP 唯一的数据增强就是随机裁剪
- Linear Probe：冻住主体参数，抽特征，只微调分类头
- 过于难的任务还得 few-shot
- 推文/视频下的 hashtag 可以作为监督信号的来源
- Clip 把原超参 temperature 优化为了可以学习的标量

伪代码：

```
# image_encoder      ResNet / Vision Transformer
# text_encoder       CBOW / Text Transformer
I[n, h, w, c]        minibatch of aligned images   (224 224 3)
T[n, l]              minibatch of aligned texts
W_i[d_i, d_e]        learned proj of image to embed
W_t[d_t, d_e]        learned proj of text to embed
t                    learned temperature
```

No.

Date

```
# extract feature representations of each modality
I_f = image_encoder(I)
T_f = text_encoder(T)

# joint multimodal embedding   投影(单模态 -> 多模态)
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arrange(n)   正样本位于对角线,与MOCO只在0号(第一位)正样本不同
                         ground truth
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, label, axis)
loss = (loss_i + loss_t)/2
```

局限性

- 想要达到专业领域SOTA水平,还需扩大1000倍 (Res 50只是基线)
- 细分类(如数个数,异常分类)情况下性能不好
- 泛化性在不属于0.4 billion的样本中很差,如MNIST
- 还得先给分类,CLIP再判断 (对比式和生成式的目标函数能否融合?)

self-supervision / self-training 或许能提高数据利用率

自监督　　伪标签

No.

Date

、训练时重复使用已有监督训练数据集，可能有偏见

· 有时 one/ few shot 反而不如 zero-shot 好
  复杂概念无法用语言描述

意义：

· 打破了固定标签种类的范式，只需搜集图片-文本的配对
  用无监督的方式预测其相似性 / 生成
  方便做推理、zero-shot

新意度：100

有效性：100

问题大小：100 （灵活，多领域，高效）

No.

Date

Convolutional Neural Networks

Edge detection.

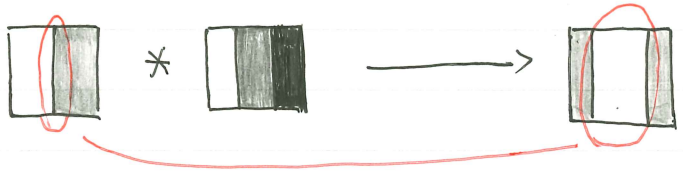
Vertical :

1	0	-1
1	0	-1
1	0	-1

Horizontal:

1	1	1
0	0	0
-1	-1	-1

filter kernel



1	0	-1
2	0	-2
1	0	-1

Sobel filter

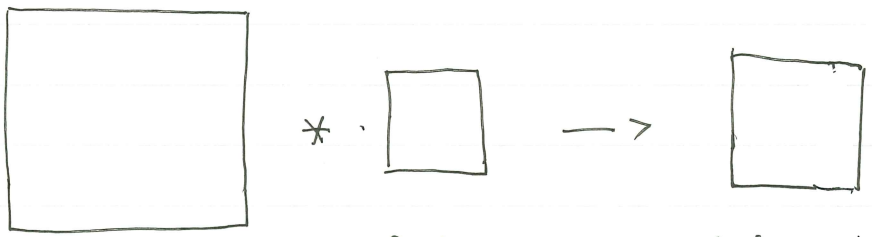
3	0	-3
10	0	-10
3	0	-3

Scharr filter

w_1	w_2	...
.	.	.
.	.	.

learnable ?

with backpropogation



$n \times n$

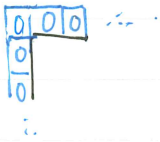
$f \times f$

$n-f+1 \cdot n-f+1$

No.

Date

Padding



$n-f+1 \longrightarrow n+2p-f+1$

Valid convolution : no padding $n-f+1$

Same convolution: (padding) Soutput = Sinput $n+2p-f+1=n \Rightarrow p=\frac{f-1}{2}$

By convention, f is always odd $\longrightarrow$ To align with "central pixel"

Strided convolution

output: $\frac{n+2p-f}{s}+1 \cdot \frac{n+2p-f}{s}+1$

convolution in math textbook

Convolution in DL is Cross-correlation in Math

For convolution layer l

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

Input: $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n_{H/W}^{[l]} = \frac{n^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

$n_c^{[l]}$ = number of filters

Each filter: $f^{[l]} \cdot f^{[l]} \cdot n_c^{[l-1]}$

Activation: $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \cdot f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1,1,1,n_c^{[l]})$

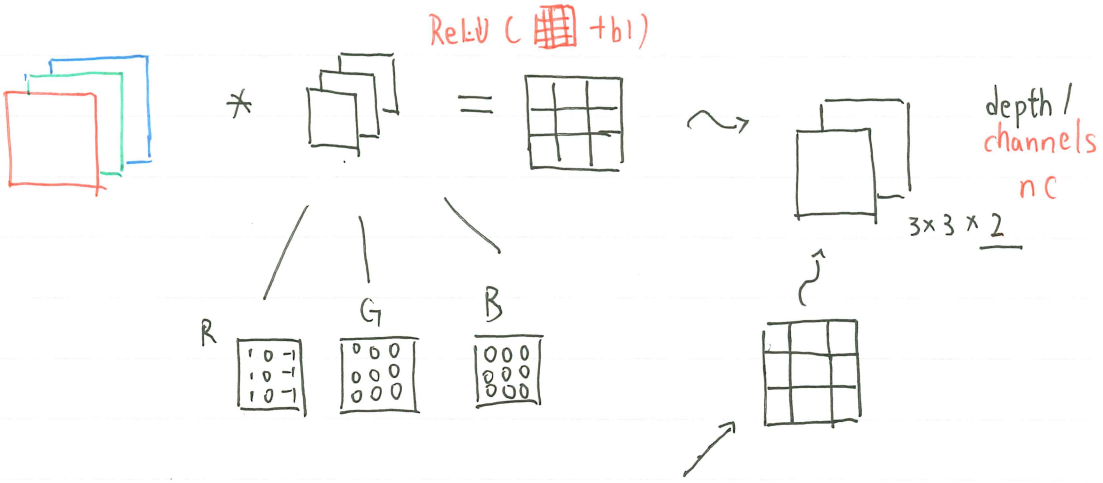
Type of layers

1. Conv
2. Pool
3. FC

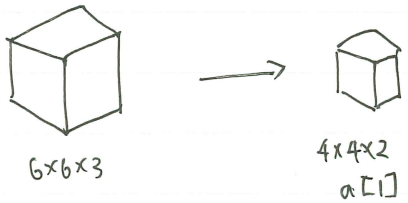
No.

Date

Convolution on volumn.



- We can have multiple filters focusing on different features.



N parameters remain same no matter how big input image is
— less likely to overfit

Max Pooling
Hyper Para: f s nothing to learn
2 2

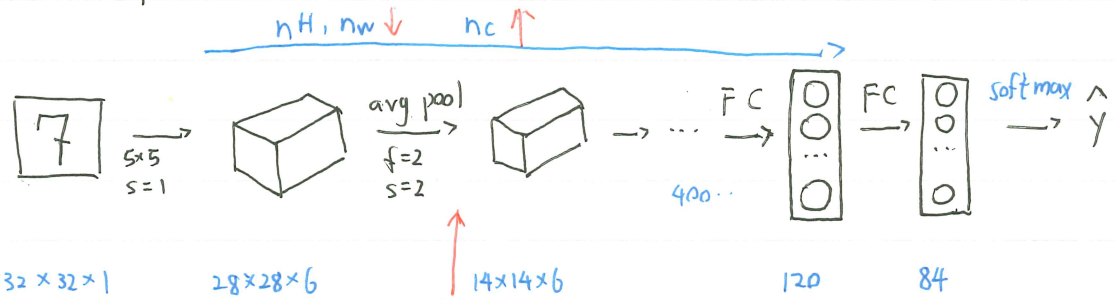
Average pooling (less likely to use?
Used when very deep

No.

Date

Le Net - 5

60k parameters.

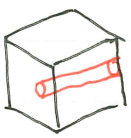


Used to have a non-linear (sigmoid / tanh.

Conv pool .... fc fc output

Alex Net. max pool

- Similar to LeNet, but much bigger
- ReLU (solve gradient diffusion of sigmoid in deeper nets)
- Multiple GPUs
- Local Response Normalization



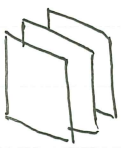
Normalization

- Utilize Dropout

VGG - 16 (Simpler, less hp)

138 M parameters

CONV = 3×3 filter, s = 1. same, MAX POOL = 2×2, s = 2



[CONV 64] → POOL → CONV 128 ×2, 256 ×2, 512 ×3 → POOL → FC 4096 → FC 4096 → Softmax 1000

512 big enough

Every step, as we go deeper

nw, nh /= 2

nc *= 2

very systematic uniform structure.

No.

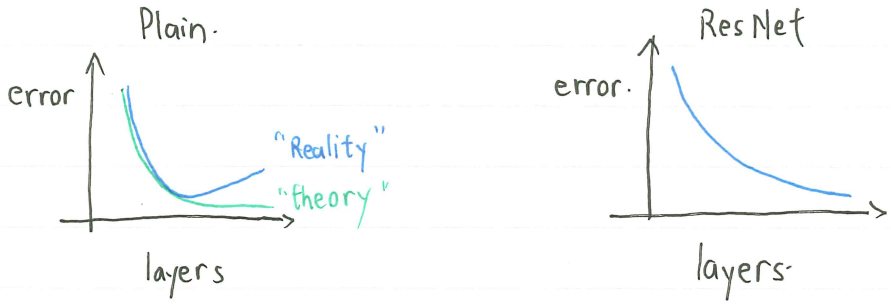
Date

Res Net Residual block.

short cut / skip connection

$a^{[i]}$ → linear → ReLU → $a^{[i+1]}$ → linear → ReLU → $a^{[i+2]}$

main path.



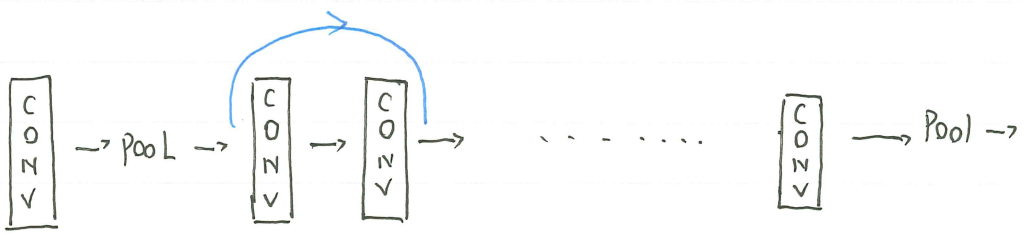
Allow for training deeper neural networks

Why Residual Network works

Normally $a^{[i]} = a^{[i+2]}$, it won't affect overall performance

More chances to choose correct parameter to improve.

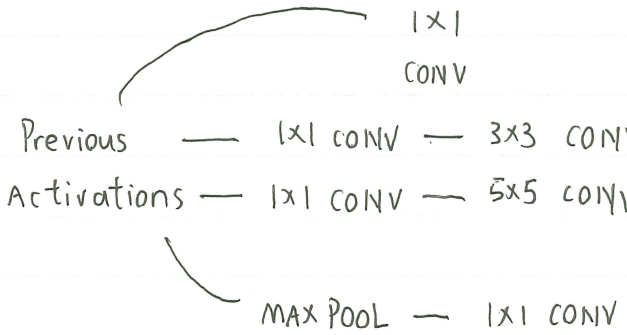
* if dimensions of $a^{[i]}$ and $a^{[i+2]}$ are different, add a weight matrix



No.

Date

Inception block



- To prevent overfitting, we also add some softmax branches
  (like a regularizing effect)

Depthwise Convolution

$6\times6\times3 \quad * \quad 3\times3\times n_c \longrightarrow 4\times4\times3$

Cost = filter params × filter positions × # of filters

Pointwise Convolution

$4\times4\times3 \quad * \quad 1\times1\times n_c \longrightarrow 4\times4\times n_c'$

Cost is similiar to above.

Depthwise Separable Convolution

Depthwise Pointwise.

most computational efficient compared with normal convolution

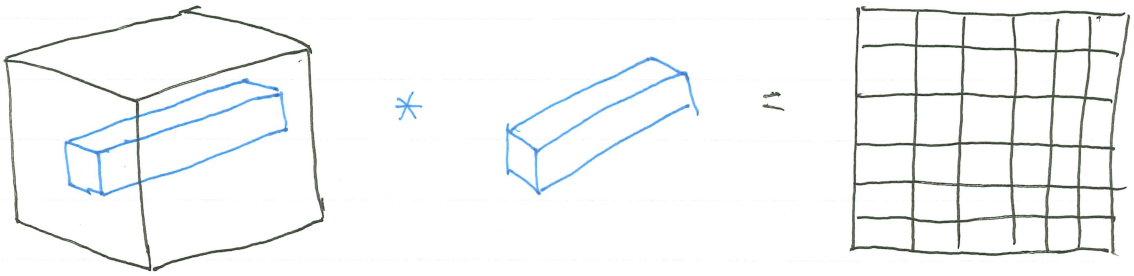
$\frac{1}{n_c'} + \frac{1}{f^2}$ more efficient

5

No.

Date

1×1 convolution: (network in network)

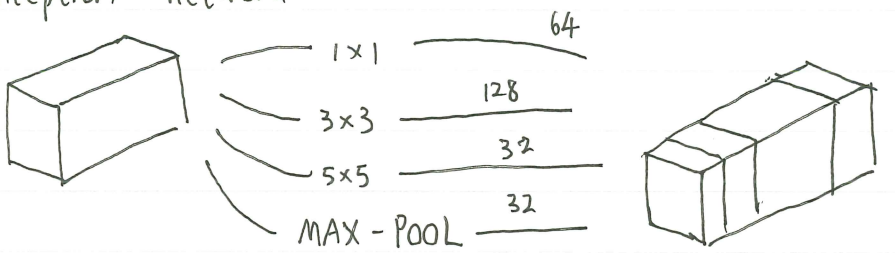


6×6×32 1×1×32 6×6×#filters

similiar to a FC layer that applies to each positions

Adjust the channel. (升/降维特征矩阵)

Inception network.

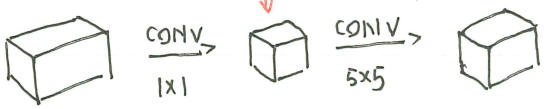


n·n·192 n·n·256

No need to pick kernel size, let it learn

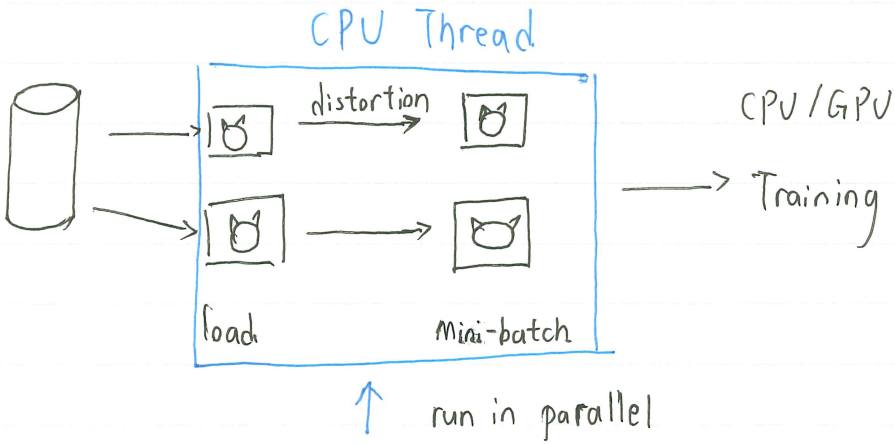
Problem: computational cost

"bottleneck layer"



Computation 120 m → 12.4 m

If bottleneck layer is implemented properly, you can shrink down representation size significantly while doesn't seem to hurt performance



Two sources of knowledge

- Labeled data
- Hand engineered features / network architecture

Tip. —— Ensembling

- Train several networks independently and average their outputs

Multi-crop at test time.

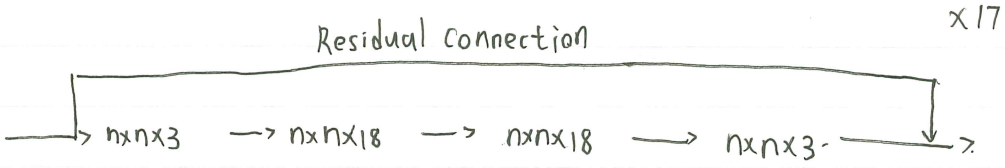
- Run classifier on multiple versions of test images and average

Reduce speed · better not use it when building products results

No.

Date

Mobile Net V2 Bottle Neck



Expansion Depthwise Pointwise.

(Projection due to memory limit)

allow for learning richer and more complex functions

Efficient Net: How to choose. r (resolution), d (depth) and w (width)?

Transfer Learning

* Freeze previous parameters. modify the last layer
* Use others' parameters as initialization.
  . More data —> less frozen layers

Data Augmentation.

- Mirroring
- Random Cropping
- Rotation
- Shearing
- Local Warping 局部约束
- Color shifting

PCA color augmentation: : more changes on R·B · less on G

No.

Date

No.

Date