# Week 5: Convolutional Neural Networks - 2

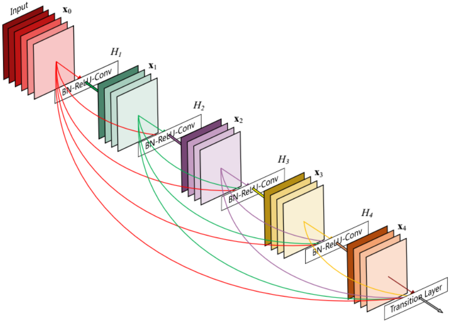
Instructor: Ruixuan Wang
wangruix5@mail.sysu.edu.cn

School of Data and Computer Science
Sun Yat-Sen University

28 March, 2019

# From ResNets to DenseNets

- DenseNet: connect all layers directly with each other

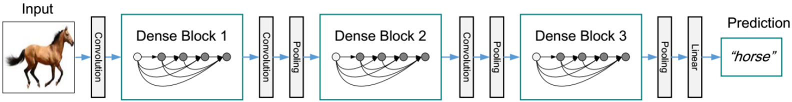



- Differ from Resnet: not sum, but **concatenate** feature maps
- Encourages **feature reuse** throughout the network
- Can ensure maximum information flow between layers

figure from Huang, Liu, van der Maaten, Weinberger, "Densely connected convolutional networks", CVPR 2017

# DenseNets

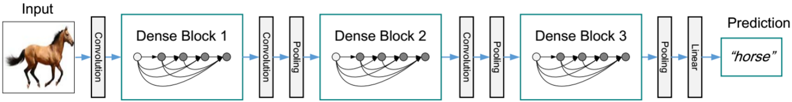
- Use transition layers to change size of feature maps




- Training: easily back-propagate errors via skip connections
- Each layer can have fewer (e.g., 12) kernels, and still sufficient to obtain state-of-the-art results
- Reason: each layer has access to all preceding feature-maps in its block, and therefore to the network's 'collective knowledge'

# DenseNets

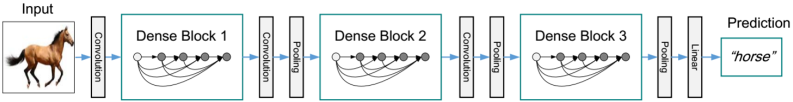
- Use transition layers to change size of feature maps




- Training: easily back-propagate errors via skip connections
- Each layer can have fewer (e.g., 12) kernels, and still sufficient to obtain state-of-the-art results
- Reason: each layer has access to all preceding feature-maps in its block, and therefore to the network's 'collective knowledge'

# DenseNets

- Use transition layers to change size of feature maps



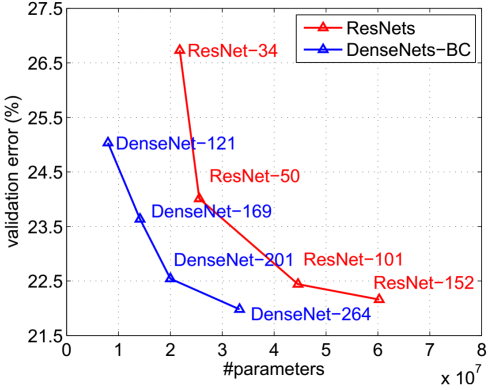

- Training: easily back-propagate errors via skip connections
- Each layer can have fewer (e.g., 12) kernels, and still sufficient to obtain state-of-the-art results
- Reason: each layer has access to all preceding feature-maps in its block, and therefore to the network's 'collective knowledge'

# DenseNets (cont')

| Layers | Output Size | DenseNet-121 | DenseNet-169 | DenseNet-201 |
|---|---|---|---|---|
| Convolution | $112 \times 112$ | $7 \times 7$ conv, stride 2 | | |
| Pooling | $56 \times 56$ | $3 \times 3$ max pool, stride 2 | | |
| Dense Block (1) | $56 \times 56$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ |
| Transition Layer (1) | $56 \times 56$ | $1 \times 1$ conv | | |
| | $28 \times 28$ | $2 \times 2$ average pool, stride 2 | | |
| Dense Block (2) | $28 \times 28$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ |
| Transition Layer (2) | $28 \times 28$ | $1 \times 1$ conv | | |
| | $14 \times 14$ | $2 \times 2$ average pool, stride 2 | | |
| Dense Block (3) | $14 \times 14$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$ |
| Transition Layer (3) | $14 \times 14$ | $1 \times 1$ conv | | |
| | $7 \times 7$ | $2 \times 2$ average pool, stride 2 | | |
| Dense Block (4) | $7 \times 7$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$ |
| Classification Layer | $1 \times 1$ | $7 \times 7$ global average pool | | |
| | | 1000D fully-connected, softmax | | |

- Tens of conv layers per dense block
- Use $1 \times 1$ conv layer to reduce number of feature maps

# DenseNets (cont')




- Fewer parameters (and computation) for similar performance
- Less prone to overfitting (due to fewer parameters)

# More ResNet variations

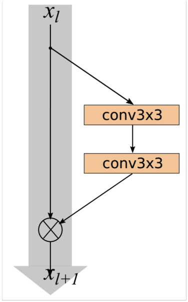
Sometimes, we may think differently for innovation!

# More ResNet variations

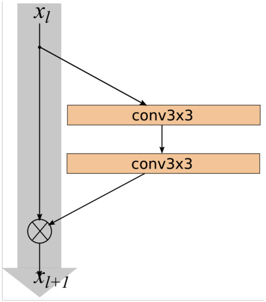
Sometimes, we may think differently for innovation!

How about not making network deeper, but wider?

# Wide ResNet




Basic ResNet layer

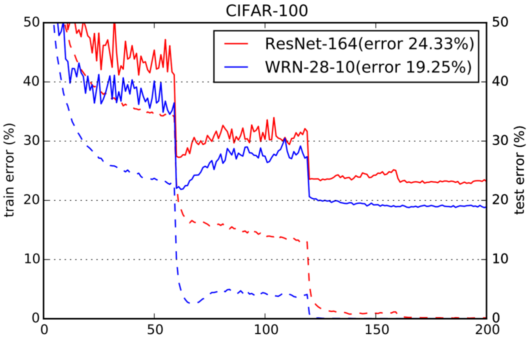



Wide ResNet layer

- Wide Resnet: more kernels in each layer, with fewer layers

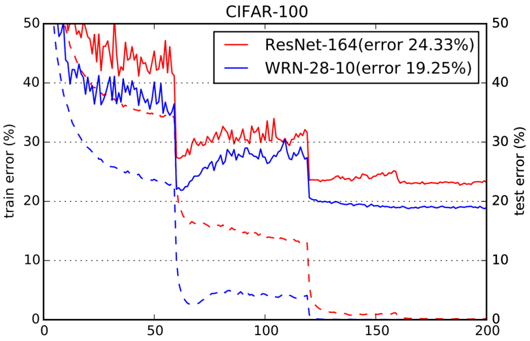
figure from Zagoruyko, Komodakis, “Wide residual networks”, BMVC 2016

# Wide ResNet (cont')




- 'WRN-28-10': 28 conv layers, 10 times original kernel number
- Widening improves performance
- So, ResNet works not due to extreme depth: 'skip connection'

# Wide ResNet (cont')



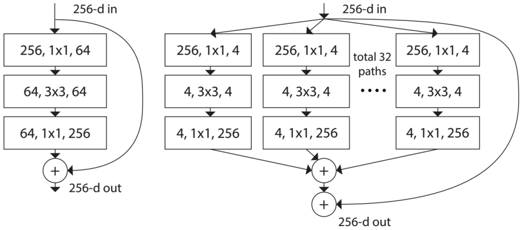

- 'WRN-28-10': 28 conv layers, 10 times original kernel number
- Widening improves performance
- So, ResNet works not due to extreme depth: 'skip connection'

# More ResNet variations

Besides deeper and wider networks ...

# ResNeXt: besides deeper and wider

- ResNeXt: divide ResNet block into smaller transformations, then aggregate.

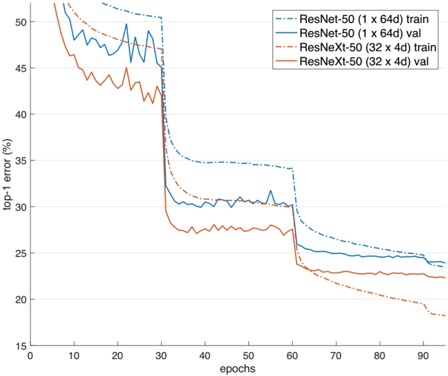



- Left: ResNet block; Right:ResNeXt block with **cardinality** 32
- Each layer: (# input channels, filter size, # output channels)
- Two blocks have similar number of parameters

Figures and tables from Xie, Girshick, Dollar, Tu, He, "Aggregated residual transformations for deep neural networks", CVPR 2017

# ResNeXt (cont')

- Comparison on ImageNet-1K dataset




- ResNeXt performs better than ResNet, with similar complexity
- Lower training error indicates more powerful feature learning

# ResNeXt (cont')

- Comparison on ImageNet-1K dataset

| | setting | top-1 err (%) | top-5 err (%) |
|---|---|---|---|
| *1× complexity references:* | | | |
| ResNet-101 | 1 × 64d | 22.0 | 6.0 |
| ResNeXt-101 | 32 × 4d | 21.2 | 5.6 |
| *2× complexity models follow:* | | | |
| ResNet-**200** [14] | 1 × 64d | 21.7 | 5.8 |
| ResNet-101, wider | 1 × **100**d | 21.3 | 5.7 |
| ResNeXt-101 | **2** × 64d | 20.7 | 5.5 |
| ResNeXt-101 | **64** × 4d | **20.4** | **5.3** |

- Increasing cardinality is better than increasing depth and width

# ResNeXt (cont')

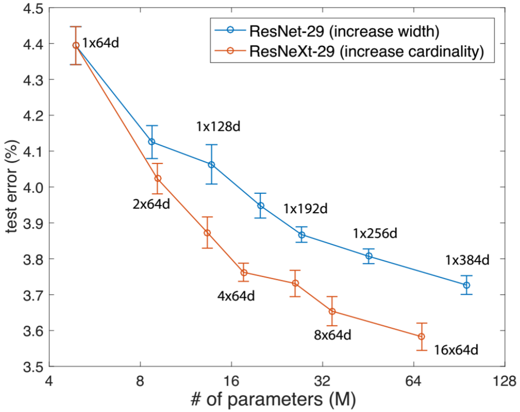
- Comparison on ImageNet-1K dataset

| | setting | top-1 err (%) | top-5 err (%) |
|---|---|---|---|
| *1× complexity references:* | | | |
| ResNet-101 | 1 × 64d | 22.0 | 6.0 |
| ResNeXt-101 | 32 × 4d | 21.2 | 5.6 |
| *2× complexity models follow:* | | | |
| ResNet-**200** [14] | 1 × 64d | 21.7 | 5.8 |
| ResNet-101, wider | 1 × **100**d | 21.3 | 5.7 |
| ResNeXt-101 | **2** × 64d | 20.7 | 5.5 |
| ResNeXt-101 | **64** × 4d | **20.4** | **5.3** |

- Increasing cardinality is better than increasing depth and width

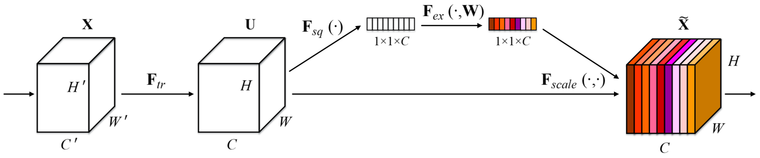
# ResNeXt (cont')

- Comparison on CIFAR-10 dataset

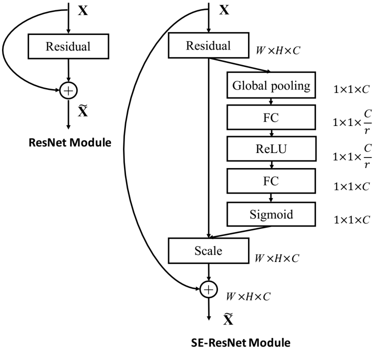



- Again: increasing cardinality is more effective than width

# SENet: Squeeze-and-Excitation Networks



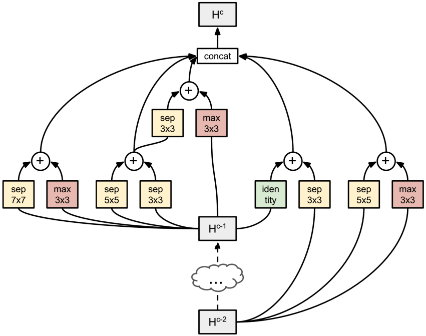

- model channel dependencies
- control channel excitation
- plug-in operator
- SENet $=$ SE-ResNeXt152
- won 2017 ImageNet contest, with $2.25\%$ top-5 error

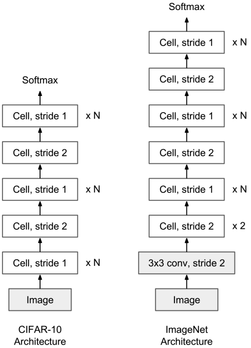



Figures from Hu, Shen, Sun, "Squeeze-and-Excitation Networks", CVPR 2018

# PNASNet: progressive neural architecture search

- All model architectures above are designed by humans.
- Recently: AutoML techniques for neural architecture search



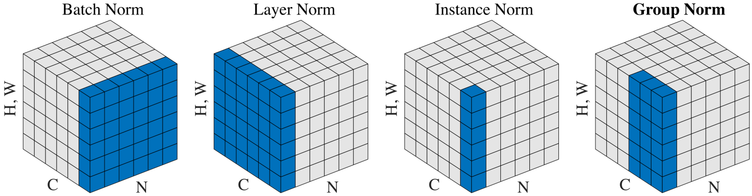

Note: ‘sep’ for ‘depthwise separable convolution’

- PNASNet outperforms almost all hand-crafted models

Figures from Liu, Zoph, Neumann, Shlens, Hua, Li, Fei-Fei, Yuille, Huang, Murphy, “Progressive Neural Architecture Search”, ECCV 2018

# Before Next...Group Normalization

- Batch normalization not work well for small batch size
- Group normalization (GN): divide channels into groups, then normalize within each group; works for single data.



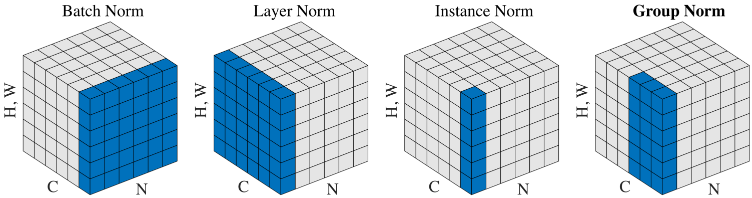

Note (in rightmost figure): 2 groups, 3 channels per group

- Layer Norm (LN) and Instance Norm (IN) are special GNs
- LN and IN not work well for visual recognition tasks

Figure from Wu, He, "Group normalization", ECCV 2018

# Before Next...Group Normalization

- Batch normalization not work well for small batch size
- Group normalization (GN): divide channels into groups, then normalize within each group; works for single data.




Note (in rightmost figure): 2 groups, 3 channels per group

- Layer Norm (LN) and Instance Norm (IN) are special GNs
- LN and IN not work well for visual recognition tasks

Figure from Wu, He, "Group normalization", ECCV 2018

# Next...

Let's see a few applications of CNN models!

# Face verification or identification

- How do you recognize whose face it is?

Who is this?

Dataset ($>10^6$) with IDs

# Face verification or identification

- How do you recognize whose face it is?

Who is this?

Dataset ($>10^6$) with IDs

- Probably: compare this face with many familiar faces in brain

# Face verification (cont')

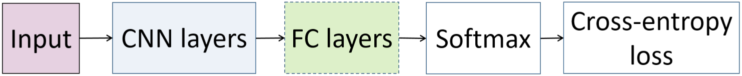
- Consider it as an image classification problem?

Input → CNN layers → FC layers → Softmax → Cross-entropy loss

- Issue 1: not comparing face similarity/distance directly
- Issue 2: too many parameters to learn
  e.g. for last layer: 100 features * 1,000,000 classes
- Issue 3: difficult to collect many faces per person
- Instead: FaceNet

# Face verification (cont')

- Consider it as an image classification problem?




- Issue 1: not comparing face similarity/distance directly
- Issue 2: too many parameters to learn
  e.g. for last layer: 100 features * 1,000,000 classes
- Issue 3: difficult to collect many faces per person
- Instead: FaceNet

# Face verification (cont')

- Consider it as an image classification problem?

Input → CNN layers → FC layers → Softmax → Cross-entropy loss

- Issue 1: not comparing face similarity/distance directly
- Issue 2: too many parameters to learn
  e.g. for last layer: 100 features * 1,000,000 classes
- Issue 3: difficult to collect many faces per person
- Instead: FaceNet

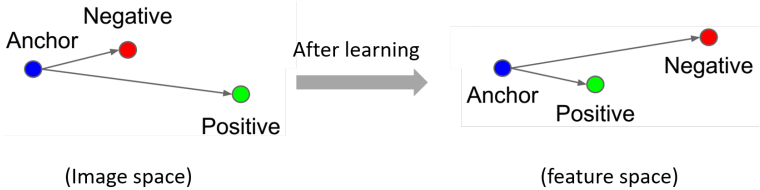
Input → CNN layers → FC layers → L2 norm → Triplet loss

# FaceNet for face verification

- FaceNet: a convolutoinal network to transform images into a low-dimensional (i.e., 128) feature space in which face images of same identity are closer than images of different identities.

# FaceNet for face verification

- FaceNet: a convolutoinal network to transform images into a low-dimensional (i.e., 128) feature space in which face images of same identity are closer than images of different identities.




- ‘anchor’ and ‘positive’ images have same identity
- ‘anchor’ and ‘negative’ images have different identity

Figure here and evaluation results below from Schroff, Kalenichenko, Philbin, “FaceNet: a unified embedding for face recognition and clustering”, CVPR 2015

# Triplet loss for FaceNet

- To ensure that an image $\mathbf{x}_i^a$ (anchor) is closer to every other images $\mathbf{x}_i^p$ (positive) of the same person than to any image $\mathbf{x}_i^n$ of any other person, we hope

$$\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha \quad < \quad \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2$$
$$\forall \; (\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) \in \mathcal{T}$$

  $\mathcal{T}$: training triplets $\{(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)\}_{i=1}^N$; $\alpha$: positive constant
  $\mathbf{f}(\cdot)$: feature representation of input via CNN model

- So, the loss over one triplet $(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)$ can be designed as

$$l(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) = \begin{cases} 0, \\ \quad \text{if } \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha < \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2 \\ \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2, \\ \quad \text{if } \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha \geq \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2 \end{cases}$$

# Triplet loss for FaceNet

- To ensure that an image $\mathbf{x}_i^a$ (anchor) is closer to every other images $\mathbf{x}_i^p$ (positive) of the same person than to any image $\mathbf{x}_i^n$ of any other person, we hope

$$\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha \quad < \quad \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2$$
$$\forall\ (\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) \in \mathcal{T}$$

  $\mathcal{T}$: training triplets $\{(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)\}_{i=1}^N$; $\alpha$: positive constant
  $\mathbf{f}(\cdot)$: feature representation of input via CNN model

- So, the loss over one triplet $(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n)$ can be designed as

$$l(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) \quad = \quad \begin{cases} 0, & \\ \quad \text{if } \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha < \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2 & \\ \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2, & \\ \quad \text{if } \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha \geq \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2 & \end{cases}$$

# Triplet loss for FaceNet (cont')

- The above loss can be abbreviated as

$$l(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) = \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

- Triplet loss: on the whole training dataset

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

$\boldsymbol{\theta}$: model parameters

# Triplet loss for FaceNet (cont')

- The above loss can be abbreviated as

$$l(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) = \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

- Triplet loss: on the whole training dataset

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

$\boldsymbol{\theta}$: model parameters

# Triplet loss for FaceNet (cont')

- The above loss can be abbreviated as

$$l(\mathbf{x}_i^a, \mathbf{x}_i^p, \mathbf{x}_i^n) \quad = \quad \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

- Triplet loss: on the whole training dataset

$$L(\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{N} \left[\|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^p)\|^2 + \alpha - \|\mathbf{f}(\mathbf{x}_i^a) - \mathbf{f}(\mathbf{x}_i^n)\|^2\right]_+$$

$\boldsymbol{\theta}$: model parameters

- Output of FaceNet is not a class label or probability, but a 128-dim feature vector $\mathbf{f}(\mathbf{x})$

# FaceNet: evaluation

- Training set generation: select 'hard' triplets which have larger triplet loss!
- CNN models: VGG-like or GoogleNet, $15 \sim 22$ layers

- LFW dataset: Labelled Faces in the Wild
  13233 images; 5749 people; 1680 people with $\geq 2$ images
- In training: used other labelled data set.
- Verification rule: given two face images $\mathbf{x}_i$ and $\mathbf{x}_j$, if $\|\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)\|^2 < \tau$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered 'same'.
- Threshold $\tau = 1.24 \sim 1.26$ was experimentally decided

- Verification accuracy: $99.63\% \pm 0.09\%$, better than humans!

# FaceNet: evaluation

- Training set generation: select 'hard' triplets which have larger triplet loss!
- CNN models: VGG-like or GoogleNet, $15 \sim 22$ layers

- LFW dataset: Labelled Faces in the Wild
13233 images; 5749 people; 1680 people with $\geq 2$ images
- In training: used other labelled data set.
- Verification rule: given two face images $\mathbf{x}_i$ and $\mathbf{x}_j$, if $\|\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)\|^2 < \tau$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered 'same'.
- Threshold $\tau = 1.24 \sim 1.26$ was experimentally decided

- Verification accuracy: $99.63\% \pm 0.09\%$, better than humans!

# FaceNet: evaluation

- Training set generation: select 'hard' triplets which have larger triplet loss!
- CNN models: VGG-like or GoogleNet, $15 \sim 22$ layers

- LFW dataset: Labelled Faces in the Wild
  13233 images; 5749 people; 1680 people with $\geq 2$ images
- In training: used other labelled data set.
- Verification rule: given two face images $\mathbf{x}_i$ and $\mathbf{x}_j$, if $\|\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)\|^2 < \tau$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered 'same'.
- Threshold $\tau = 1.24 \sim 1.26$ was experimentally decided

- Verification accuracy: $99.63\% \pm 0.09\%$, better than humans!

# FaceNet: evaluation

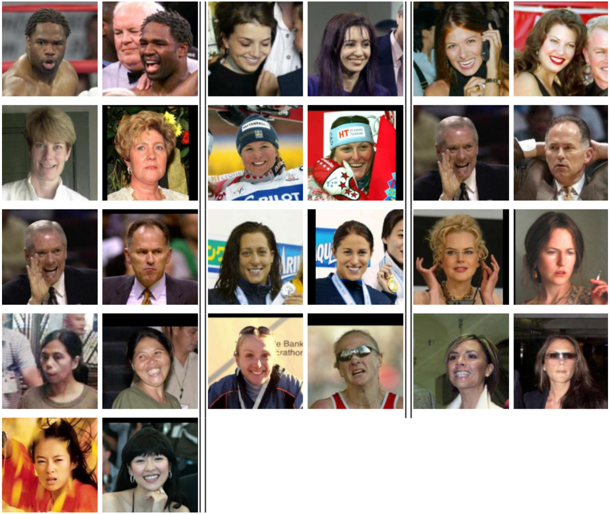
- Training set generation: select 'hard' triplets which have larger triplet loss!
- CNN models: VGG-like or GoogleNet, $15 \sim 22$ layers

- LFW dataset: Labelled Faces in the Wild
  13233 images; 5749 people; 1680 people with $\geq 2$ images
- In training: used other labelled data set.
- Verification rule: given two face images $\mathbf{x}_i$ and $\mathbf{x}_j$, if $\|\mathbf{f}(\mathbf{x}_i) - \mathbf{f}(\mathbf{x}_j)\|^2 < \tau$, then $\mathbf{x}_i$ and $\mathbf{x}_j$ are considered 'same'.
- Threshold $\tau = 1.24 \sim 1.26$ was experimentally decided

- Verification accuracy: $99.63\% \pm 0.09\%$, better than humans!

# FaceNet: evaluation (cont')

- Do you think each pair is from same person?



# FaceNet: evaluation (cont')

- Do you think each pair is from same person?

# FaceNet: evaluation (cont')

- Do you think each pair is from same person?


- Each pair is from different persons
- previous page: each pair from same person

# FaceNet: evaluation (cont')

- Do you think each pair is from same person?

- Each pair is from different persons
- previous page: each pair from same person
- These pairs are ALL the mistakes made by FaceNet!

# Style transfer

- Generate a new image $\mathbf{x}$ which has style of image $\mathbf{a}$ and content of image $\mathbf{p}$

Style image $\mathbf{a}$

Content image $\mathbf{p}$

Style transfer

Style-transferred image $\mathbf{x}$

Pictures from https://harishnarayanan.org/writing/artistic-style-transfer/

# Style transfer: loss function

- Idea: design and minimize a loss function

$$\mathcal{L}(\mathbf{x}; \mathbf{a}, \mathbf{p}, \boldsymbol{\theta}) = \mathcal{L}_{content}(\mathbf{p}, \mathbf{x}) + \lambda \mathcal{L}_{style}(\mathbf{a}, \mathbf{x})$$

such that

$\mathcal{L}_{\text{content}}$ (  ,  ) $\approx 0$

$\mathcal{L}_{\text{style}}$ (  ,  ) $\approx 0$

- To be optimized is not CNN parameters, but input image $\mathbf{x}$

# Style transfer: loss function

- Idea: design and minimize a loss function

$$\mathcal{L}(\mathbf{x};\mathbf{a},\mathbf{p},\boldsymbol{\theta}) \quad = \quad \mathcal{L}_{content}(\mathbf{p},\mathbf{x}) + \lambda\,\mathcal{L}_{style}(\mathbf{a},\mathbf{x})$$

  such that

$\mathcal{L}_{\text{content}}\Big($  ,  $\Big) \approx 0$

$\mathcal{L}_{\text{style}}\Big($  ,  $\Big) \approx 0$

- To be optimized is not CNN parameters, but input image $\mathbf{x}$

## Style transfer: loss function

- Idea: design and minimize a loss function

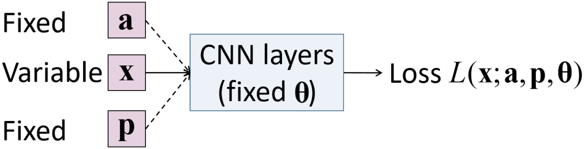
$$\mathcal{L}(\mathbf{x};\mathbf{a},\mathbf{p},\boldsymbol{\theta}) \quad = \quad \mathcal{L}_{content}(\mathbf{p},\mathbf{x}) + \lambda\,\mathcal{L}_{style}(\mathbf{a},\mathbf{x})$$

such that

$\mathcal{L}_{\text{content}}($  ,  $) \approx 0$

$\mathcal{L}_{\text{style}}($  ,  $) \approx 0$

- To be optimized is not CNN parameters, but input image $\mathbf{x}$
- How is CNN model applied here?

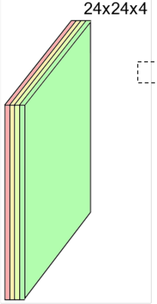


## Style transfer: loss function (cont')

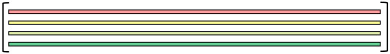
- Similar content: feature maps from conv layers should be similar between $\mathbf{x}$ and $\mathbf{p}$
- Similar style: textures in feature maps should be similar between $\mathbf{x}$ and $\mathbf{a}$

# Style transfer: loss function (cont')

- Similar content: feature maps from conv layers should be similar between $\mathbf{x}$ and $\mathbf{p}$
- Similar style: textures in feature maps should be similar between $\mathbf{x}$ and $\mathbf{a}$




Feature maps at layer $l$



$F^l$ Feature matrix of size $4 \times 576$

Gram matrix $G^l = \begin{bmatrix} G_{11} & G_{12} & G_{13} & G_{14} \\ G_{21} & G_{22} & G_{23} & G_{24} \\ G_{31} & G_{32} & G_{33} & G_{34} \\ G_{41} & G_{42} & G_{43} & G_{44} \end{bmatrix}$

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}$$

# Style transfer: loss function (cont')

- Content loss: measured by difference in feature matrix

$$\mathcal{L}_{content}(\mathbf{p}, \mathbf{x}) = \sum_{i,j} (F^l_{ij} - P^l_{ij})^2$$

where $F^l_{ij}$ is the activation of the $i$-th filter at position $j$ in layer $l$ for CNN input $\mathbf{x}$; similarly $P^l_{ij}$ for input $\mathbf{p}$.

- Style loss: measured by difference in gram matrix

$$E_l = \sum_{i,j} (G^l_{ij} - A^l_{ij})^2$$

$$\mathcal{L}_{style}(\mathbf{a}, \mathbf{x}) = \sum_l w_l E_l$$

where $G^l$ and $A^l$ are gram matrices corresponding to input $\mathbf{x}$ and $\mathbf{p}$, respectively; $w_l$ is weighting factor.

# Style transfer: loss function (cont')

- Content loss: measured by difference in feature matrix

$$\mathcal{L}_{content}(\mathbf{p}, \mathbf{x}) = \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

where $F_{ij}^l$ is the activation of the $i$-th filter at position $j$ in layer $l$ for CNN input $\mathbf{x}$; similarly $P_{ij}^l$ for input $\mathbf{p}$.

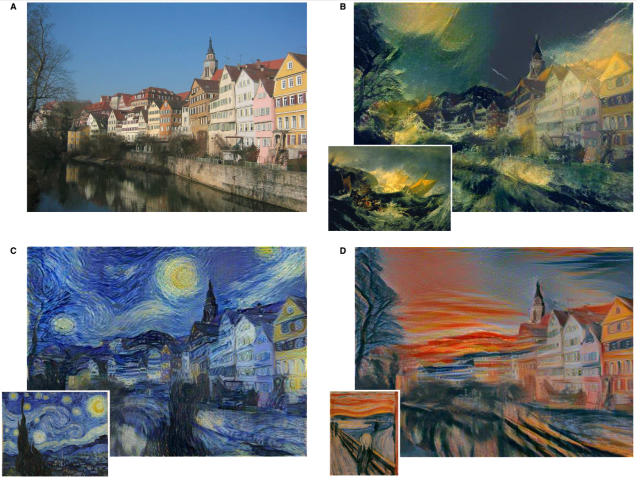
- Style loss: measured by difference in gram matrix

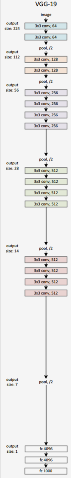
$$\begin{aligned} E_l &= \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \\ \mathcal{L}_{style}(\mathbf{a}, \mathbf{x}) &= \sum_{l} w_l E_l \end{aligned}$$

where $G^l$ and $A^l$ are gram matrices corresponding to input $\mathbf{x}$ and $\mathbf{p}$, respectively; $w_l$ is weighting factor.

# Style transfer: evaluation

- Match content on VGG layer 'conv4_2' and style on layers 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_ 1' and 'conv5_1'



# Style transfer: evaluation

- Can easily adjust the trade-off between content and style

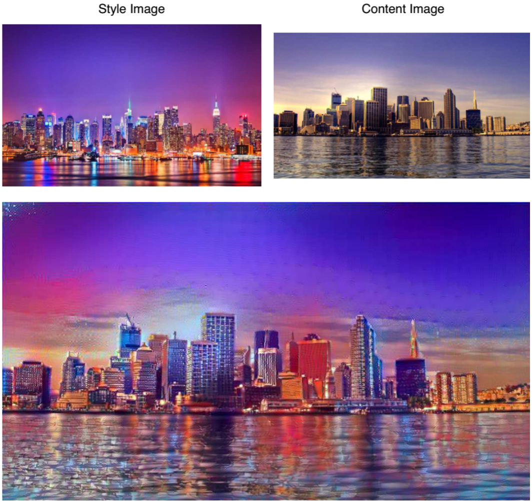
Figures from Gatys, Ecker, Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

# Style transfer: evaluation

- Lower-layer content matching kept more fine structures

# Style transfer: evaluation

- Style can be transferred from one photo to another



# Text classification

- Objective: classification of sentences or paragraph into one of predefined categories.
- E.g., sentiment analysis, news categorization

| Dataset | Label | Sample |
|---|---|---|
| Yelp P. | +1 | Been going to Dr. Goldberg for over 10 years. I think I was one of his 1st patients when he started at MHMG. Hes been great over the years and is really all about the big picture. [...] |
| Amz P. | 3(/5) | I love this show, however, there are 14 episodes in the first season and this DVD only shows the first eight. [...]. I hope the BBC will release another DVD that contains all the episodes, but for now this one is still somewhat enjoyable. |
| Sogou | "Sports" | ju4 xi1n hua2 she4 5 yue4 3 ri4 , be3i ji1ng 2008 a4o yu4n hui4 huo3 ju4 jie1 li4 ji1ng guo4 shi4 jie4 wu3 da4 zho1u 21 ge4 che2ng shi4 |
| Yah. A. | "Computer, Internet" | "What should I look for when buying a laptop? What is the best brand and what's reliable?","Weight and dimensions are important if you're planning to travel with the laptop. Get something with at least 512 mb of RAM. [..] is a good brand, and has an easy to use site where you can build a custom laptop." |

Figures and tables from Conneau, Schwenk, Le Cun, Barrault, "Very deep convolutional networks for text classification", arXiv 2017

# Text classification: data representation

- Every text (sentences/paragraphs) consists of (truncated or padded) 1014 characters
- A unique 16-dimensional vector to represent each character.
- So, input to CNN is a $16 \times 1014$ matrix
  Or: 16 channels, 'image' of size $1 \times 1014$ per channel

  Different from images: 1-dimensional 'image' or feature maps

# Text classification: data representation

- Every text (sentences/paragraphs) consists of (truncated or padded) 1014 characters
- A unique 16-dimensional vector to represent each character.

  ”abcdefghijklmnopqrstuvwxyz0123456
  789-,;.!?:'"/|_#$%^&*~`+=<>()[]{}”
- So, input to CNN is a $16 \times 1014$ matrix
  Or: 16 channels, 'image' of size $1 \times 1014$ per channel

  Different from images: 1-dimensional 'image' or feature maps

# Text classification: data representation

- Every text (sentences/paragraphs) consists of (truncated or padded) 1014 characters
- A unique 16-dimensional vector to represent each character.
  ”abcdefghijklmnopqrstuvwxyz0123456789-,;.!?:'"/|_#$%^&*~`+=<>()[]{}”
- So, input to CNN is a $16 \times 1014$ matrix
  Or: 16 channels, ‘image’ of size $1 \times 1014$ per channel

  Different from images: 1-dimensional ‘image’ or feature maps

# Text classification: data representation

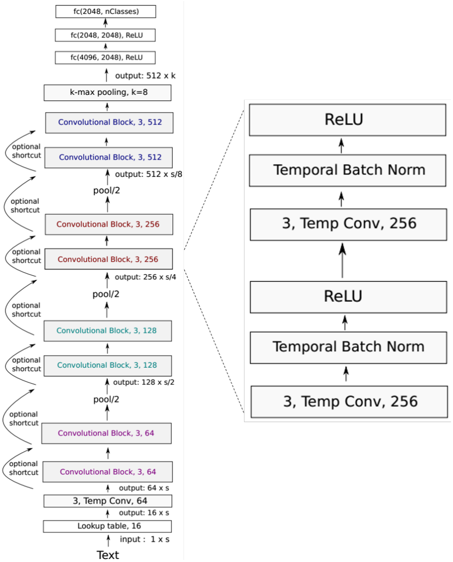
- Every text (sentences/paragraphs) consists of (truncated or padded) 1014 characters
- A unique 16-dimensional vector to represent each character.

  ”abcdefghijklmnopqrstuvwxyz0123456
  789-,;.!?:'"/|_#$%^&*~`+=<>()[]{}”
- So, input to CNN is a $16 \times 1014$ matrix

  Or: 16 channels, ‘image’ of size $1 \times 1014$ per channel

  Different from images: 1-dimensional ‘image’ or feature maps

# Text classification: model structure



# Text classification: evaluation

| Data set | #Train | #Test | #Classes | Classification Task |
|---|---|---|---|---|
| AG's news | 120k | 7.6k | 4 | English news categorization |
| Sogou news | 450k | 60k | 5 | Chinese news categorization |
| DBPedia | 560k | 70k | 14 | Ontology classification |
| Yelp Review Polarity | 560k | 38k | 2 | Sentiment analysis |
| Yelp Review Full | 650k | 50k | 5 | Sentiment analysis |
| Yahoo! Answers | 1 400k | 60k | 10 | Topic classification |
| Amazon Review Full | 3 000k | 650k | 5 | Sentiment analysis |
| Amazon Review Polarity | 3 600k | 400k | 2 | Sentiment analysis |

| Depth | Pooling | AG | Sogou | DBP. | Yelp P. | Yelp F. | Yah. A. | Amz. F. | Amz. P. |
|---|---|---|---|---|---|---|---|---|---|
| 9 | Convolution | 10.17 | 4.22 | 1.64 | 5.01 | 37.63 | 28.10 | 38.52 | 4.94 |
| 9 | KMaxPooling | 9.83 | 3.58 | 1.56 | 5.27 | 38.04 | 28.24 | 39.19 | 5.69 |
| 9 | MaxPooling | 9.17 | 3.70 | 1.35 | 4.88 | 36.73 | 27.60 | 37.95 | 4.70 |
| 17 | Convolution | 9.29 | 3.94 | 1.42 | 4.96 | 36.10 | 27.35 | 37.50 | 4.53 |
| 17 | KMaxPooling | 9.39 | 3.51 | 1.61 | 5.05 | 37.41 | 28.25 | 38.81 | 5.43 |
| 17 | MaxPooling | 8.88 | 3.54 | 1.40 | 4.50 | 36.07 | 27.51 | 37.39 | 4.41 |
| 29 | Convolution | 9.36 | 3.61 | 1.36 | 4.35 | **35.28** | 27.17 | 37.58 | **4.28** |
| 29 | KMaxPooling | **8.67** | **3.18** | 1.41 | 4.63 | 37.00 | 27.16 | 38.39 | 4.94 |
| 29 | MaxPooling | 8.73 | 3.36 | **1.29** | **4.28** | 35.74 | **26.57** | **37.00** | 4.31 |

- Test error (2nd table) on 8 datasets (1st): deeper is better

# Summary

- CNN extensions: denser, wider, more cardinality
- CNN applied to multiple domains
- CNN for face verification: triplet loss
- CNN for style transfer: separable via conv layers?
- CNN for text classification: each text as 1D ‘image’

Further reading:

- Oord et al., Wavenet: A generative model for raw audio, arXiv, 2016
- Deng et al., ArcFace: additive angular margin loss for deep face recognition, arXiv, 2018

# Comments on first assignment

- High-score reports follow points listed on Week 1's last slide
- Low-score reports miss some points, or no long summary!
- Understand the paper, then summarize using your own words
- Describe 'what' and 'how', and explain 'why'
- Just *occasionally* use formulae/figures/table
- Delete redundant and non-academic information
- Don't use bullet points; link paragraphs smoothly
- Experiments should be described in words, but concisely
- List full paper information
- Open grammar checker