Recent Advances in Deep Learning

Dr. Josif Grabocka

ISMLL, University of Hildesheim

Recent Advances in Deep Learning

Batch Normalization (BN)

Normalize activation values of each neuron h (or feature maps V):

- For a mini-batch of M instances $\mathcal{B} = h_1, h_2, \ldots, h_M$
- Mini-batch mean $\mu_{\mathcal{B}} \leftarrow \frac{1}{M}\sum_{i=1}^{M} h_i$ and variance $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{M}\sum_{i=1}^{M}(h_i - \mu_{\mathcal{B}})^2$
- Activations are Z-normalized and scaled with γ, β:

$$\hat{h}_i = \gamma \left(\frac{h_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \beta$$

- h is the post-nonlinearity activation (i.e. first ReLU than BN)

Source: Ioffe et al. 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Residual Network for Image Recognition

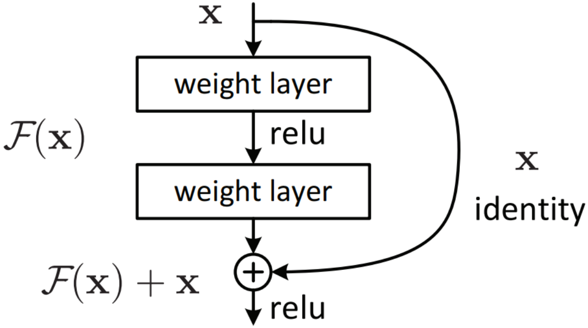

Figure 1: Residual Block, Source: He et al. 2015, Deep Residual Learning for Image Recognition

Residual Network (II)

- Define the feature map of layer ℓ as:

$$V^{(\ell)} = \max\left(0, \sum_{c=1}^{I^{(\ell)}} \sum_{m=1}^{M^{(\ell)}} \sum_{n=1}^{N^{(\ell)}} V^{(\ell-1)}_{c,x+m-1,y+n-1} K^{(\ell)}_{i,c,m,n}\right), \forall \ell \in \{1, \ldots, L\}$$

- The residual layers aggregate a specific feature map at layer $\ell + k$ with a feature map k layers ago:

$$\tilde{V}^{(\ell+k)} := \max\left(0, V^{(\ell+k)} + V^{(\ell)}\right)$$

- Typically dimensions of $V^{(\ell+k)}$ and $V^{(\ell)}$ should match, otherwise linearly project $V^{(\ell)}$ into the dimensionality of $V^{(\ell+k)}$

Residual Network Architecture

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2: ResNet Architecture, Source: He et al. 2015

ResNet Performance

Improves generalization w.r.t plain CNN without additional parameters:

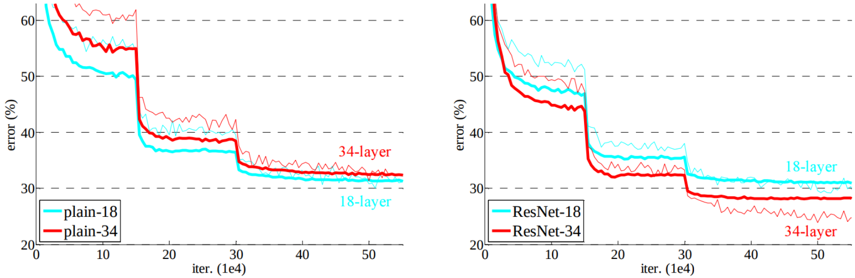

Figure 3: Residual vs plain CNNs, Source: He et al. 2015

DenseNet: A Generalization of Resnet

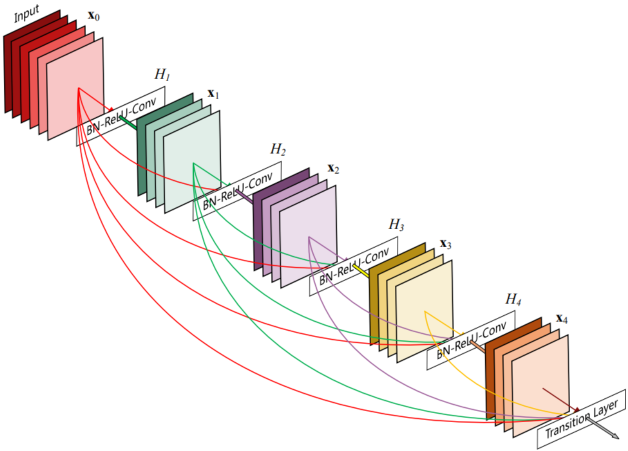

Figure 4: Residual connections from all previous layers, Source: Huang et al. 2017, Densely Connected Convolutional Networks

Dense concatenation instead of aggregation:

- Do not add previous layers as ResNet would do:

$$\tilde{V}^{(\ell)} := \max\left(0, \sum_{k=1}^{\ell} V^{(k)}\right)$$

- DenseNet instead concatenates past feature maps:

$$\tilde{V}^{(\ell)} := \left[V^{(0)}, V^{(1)}, \ldots, V^{(\ell-1)}, V^{(\ell)}\right]$$

- For L convolutional layers there are $\frac{L(L+1)}{2}$ connections
- If each convolutional layer has k kernels and the input k_0 channels
  - $k_0 + (\ell - 1)\,k$ channels as input to the ℓ-th layer
  - Yet, authors claim DenseNet's required k is way smaller than usual

DenseNet Expressiveness

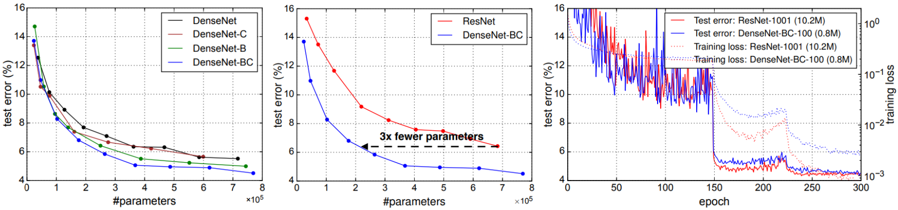

Figure 5: DenseNet is more expressive than Resnet on the CIFAR dataset; Source: Huang et al. 2017

Inception Network

- Reduce channel dimensionality via 1x1 convolutions
- Apply diverse combinations of filter sizes in one module:

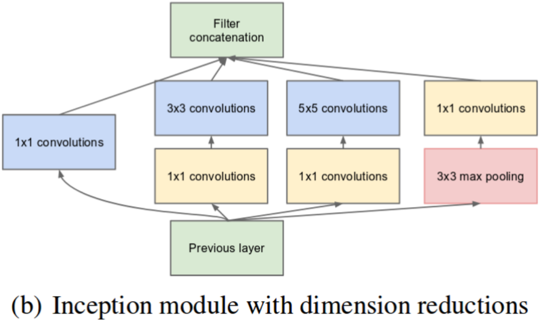

Figure 6: Inception Module, Source: Szegedy et al. 2014, Going Deeper with Convolutions

Inception+ResNet Network

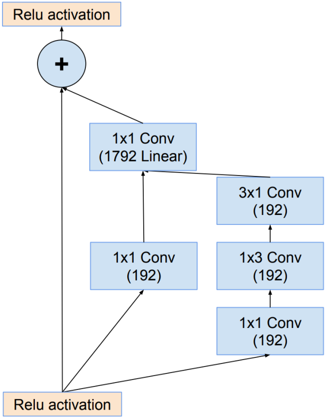

Figure 7: Inception + ResNet network module for a 8x8 grid, Source: Szegedy et al. 2016, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

Inception+ResNet Results

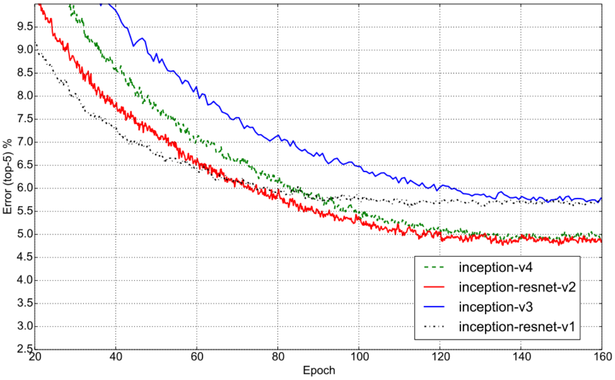

Figure 8: Top-5 error results on ImageNet, Source: Szegedy et al. 2016

Attention Mechanism

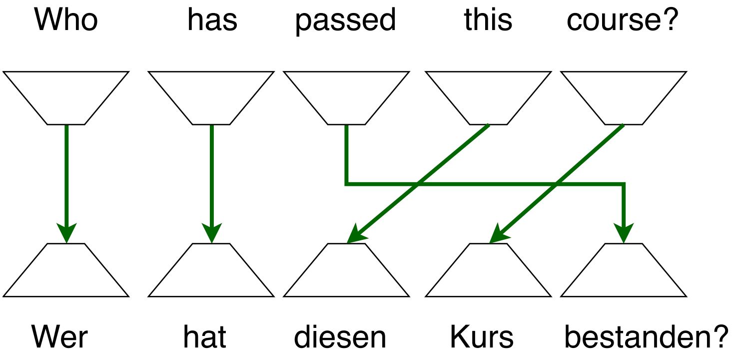

Figure 9: Translation demands "attention"

Language Encoders

- Language encoders are typically Recurrent Neural Networks that convert a word embedding $x^{(i)}$ into a latent low-rank representation $h^{(i)}$:

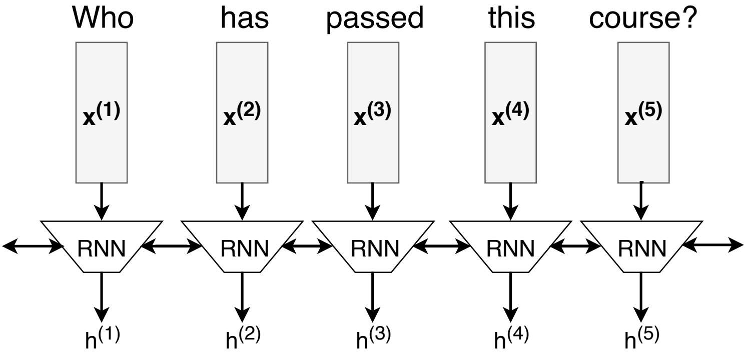

Figure 10: Encoding a sentence to a list of latent vectors

Language Decoders

- Decoder RNN: from encoded *h* into probabilities *y*

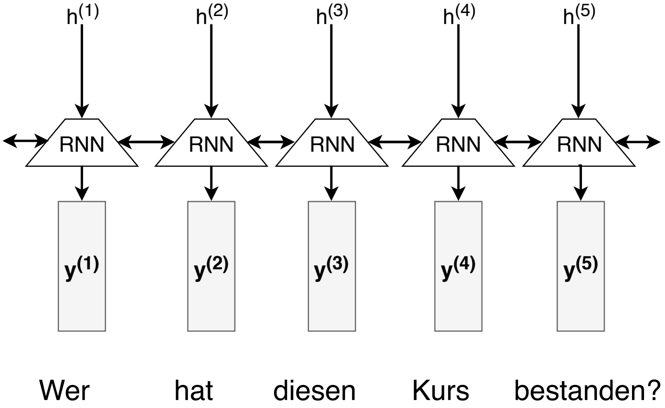

Figure 11: Decoders convert word encodings into word probabilities in the target language. **What is the problem with this model?**

Attention: Translating language A to B

- A Bi-LSTM encoder for language A with state s_A produces output:

$$h^{(i)} = \text{Bi-LSTM}^{(A)}(x^{(i)}, s_A^{(i-1)})$$

- Attention the i-th word in output sentence gets from k-th input work:

$$c^{(i)} = \sum_{k=1}^{K} \alpha_{i,k} \, h^{(k)}$$

- Finally estimate the target

$$y^{(i)} = \text{Bi-LSTM}^{(B)}(c^{(i)}, s_B^{(i-1)})$$

Attention Weights

- Attention weights are neural networks:

$$\alpha_{i,k} = \frac{e^{f_{i,k}}}{\sum_{q=1}^{K} e^{f_{i,q}}}$$

- Where $f_{i,k} = \text{NeuralNetwork}(\Delta^{(i-1)}, h^{(k)})$
  - $\Delta_{(i-1)} = s_B^{(i-1)}$ according to *Bahdanou et al. 2015, Neural Machine Translation by Jointly Learning to Align and Translate*, **or**
  - $\Delta_{(i-1)} = y^{(i-1)}$ according to *Wu et al. 2016, Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*

Google Neural Machine Translation

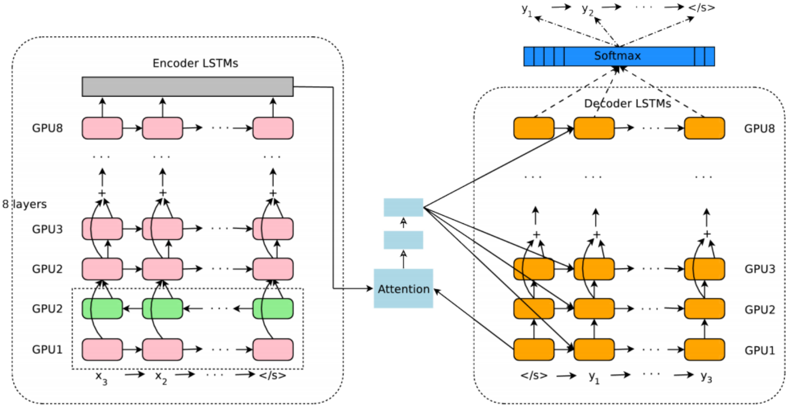

Figure 12: Google Translation System, Source: Wu et al. 2016

Neural Machine Translation Results

Table 10: Mean of side-by-side scores on production data

	PBMT	GNMT	Human	Relative Improvement
English → Spanish	4.885	5.428	5.504	87%
English → French	4.932	5.295	5.496	64%
English → Chinese	4.035	4.594	4.987	58%
Spanish → English	4.872	5.187	5.372	63%
French → English	5.046	5.343	5.404	83%
Chinese → English	3.694	4.263	4.636	60%

Figure 13: Statistical Phrase-Based vs. Neural Translation, Source: Wu et al. 2016