

Lecture 11b

**Neural Networks (cont.)**

# Outline

## Neural Networks for Images

- ▶ The Neocognitron
- ▶ Convolutional Neural Networks
- ▶ Applications

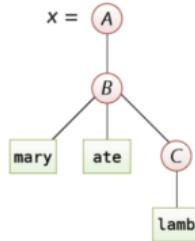
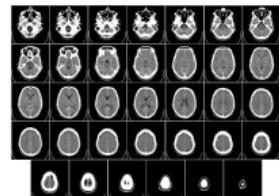
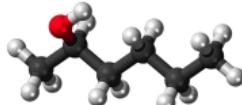
## Neural Networks for Text

- ▶ Vector representation of text
- ▶ Recursive Neural Networks (RecNNs)
- ▶ Other architectures

## Graph Neural Networks

- ▶ Structure of a Graph Neural Network
- ▶ Applications
- ▶ Relation to CNNs and RecNNs.

# Today's Lecture: NNs for Structured Data



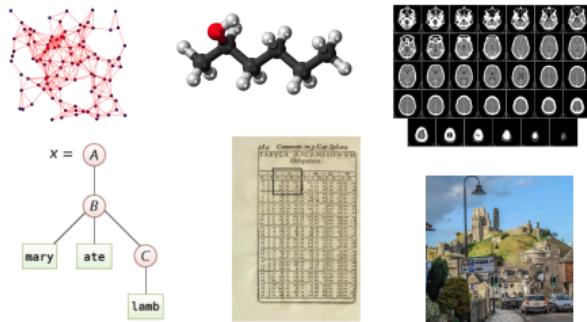
3. E. Commentarij in 3. Cap. Spherae  
TABVLA PROGRESSIONVM  
Obligationum

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400	401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500	501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600	601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800	801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900	901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000
--	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------



AAACAAATAAGTAACTAATCTTTAGGAAGAACGTTCAACCATTTGAG

# Why NNs for Structured Data?



AAACAAATAAGTAACTAATCTTTAGGAAGAACGTTCAACCATTTGAG

## Two Goals:

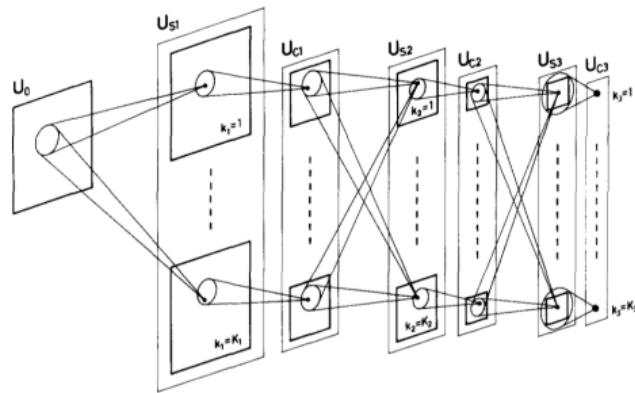
- ▶ *Technical*: Being able to process inputs that are not real-valued and cannot be represented as a fixed-size vector, e.g. text, DNA sequences.
- ▶ *Statistical*: Structured data (e.g. images) often come with specific invariances (e.g. translation invariance), that we would like the model to capture in order to generalize well.

Part 1

# Neural Networks for Images

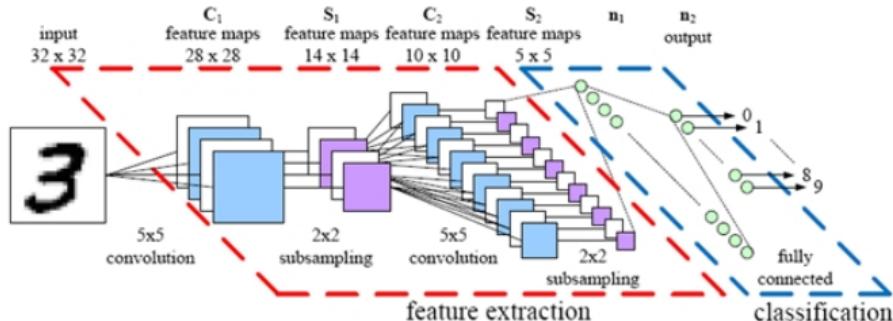
# The Neocognitron (1979)

The Neocognitron [2] is an architecture for representing images, and which is designed in a way that the produced representation is invariant to a translation of the input image.



The Neocognitron consists of an alternation of 'simple cells' (convolutions) and 'complex cells' (pooling), which progressively build the desired representation. It was a precursor of modern convolutional neural network architectures.

# Convolutional Neural Nets (1989, 1998, 2012)



- ▶ The Convolutional Neural Network (CNN) [4] is also composed of convolution and pooling layers, and comes with a way of backpropagating the error through the convolution/pooling layers.
- ▶ CNNs are nowadays state-of-the-art (by a wide margin) on many image recognition tasks (e.g. ILSVRC 2012).
- ▶ CNN training is resource-hungry. Often, people use pretrained models.

# Standard vs. Convolutional Neural Networks

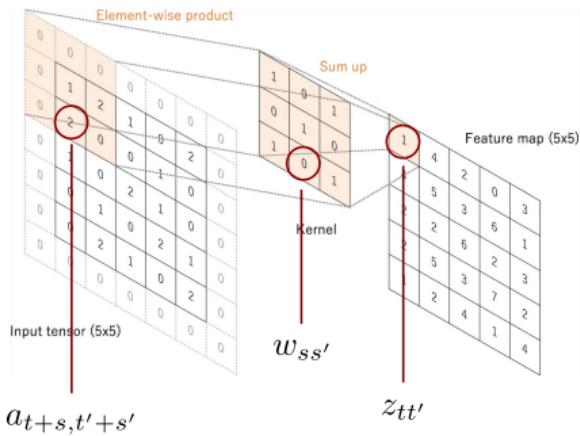
## Standard neural network:

- ▶ Neurons and weights are scalars, i.e.  $a_i, w_{ij} \in \mathbb{R}$ .
- ▶ Mapping between two layers is a sum of products:  $z_j = \sum_i w_{ij} a_i + b_j$

## Convolutional neural network:

- ▶ Neurons and weights are 2D arrays, i.e.  $\mathbf{a}^{(i)} \in \mathbb{R}^{W \times H}$ ,  $\mathbf{w}^{(ij)} \in \mathbb{R}^{W' \times H'}$ .
- ▶ Mapping between two layers is a sum of 2D cross-correlations:  
$$\mathbf{z}^{(j)} = \sum_i \mathbf{w}^{(ij)} \star \mathbf{a}^{(i)} + \mathbf{b}^{(j)}$$
- ▶ Convolution layers are interleaved with pooling layers in order to progressively reduce the spatial resolution and simultaneously increase the number of neurons.

# Understanding the CNN (1)



adapted from Yamashita et al. 2018  
Convolutional neural networks: an overview [...]

## Step 1: The Cross-Correlation

- ▶ A *cross-correlation* slides a kernel (or filter) through the whole input tensor and records responses at each location.

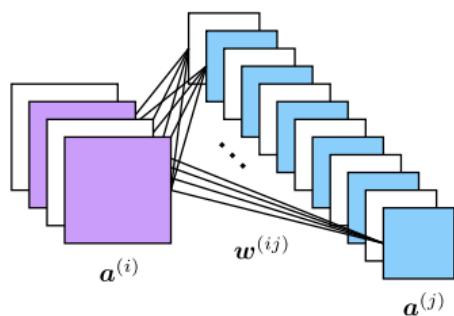
$$z_{tt'} = [w * a]_{tt'}$$

$$= \sum_{s=0}^2 \sum_{s'=0}^2 w_{ss'} \cdot a_{t+s, t'+s'}$$

- ▶ Zero-padding can be applied to the input signal in order to produce an output signal of same size as the input.

# Understanding the CNN (2)

## Step 2: The Convolution Layer

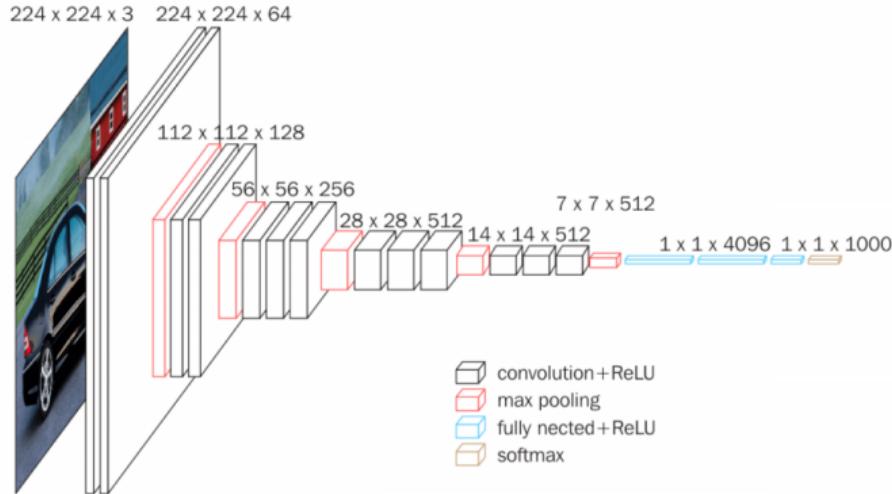


- ▶ A *convolution layer* has its output given by a sum of cross-correlations associated to each incoming neurons:

$$a^{(j)} = g\left(\sum_i w^{(ij)} * a^{(i)}\right)$$

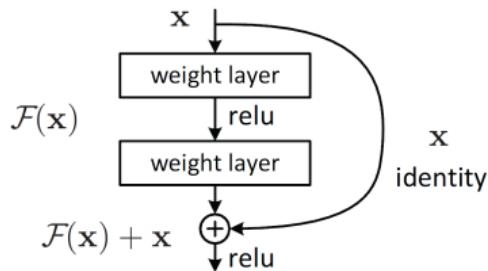
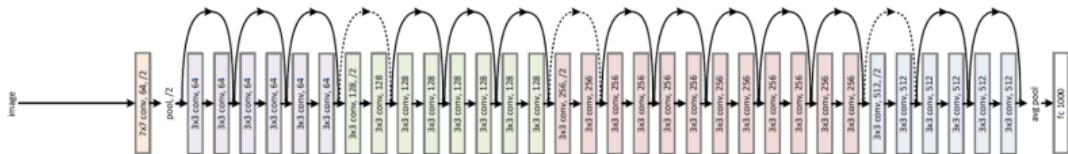
- ▶ The number of trainable parameters in a convolution layer is given by (# input neurons  $\times$  # output neurons  $\times$  filter size).

# The VGG-16 Architecture



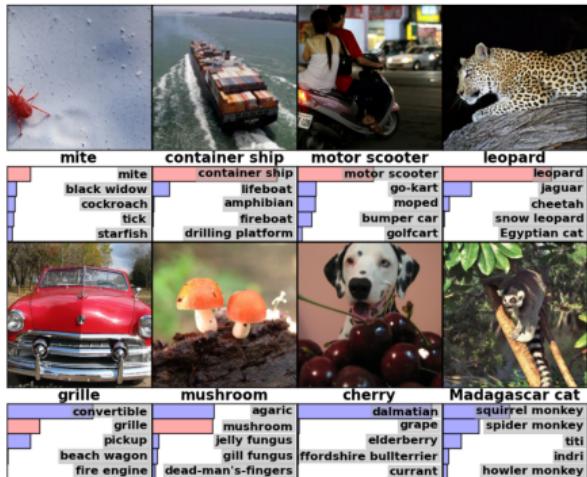
- ▶ Multiple blocks of two or three consecutive convolutions (each with kernel size  $3 \times 3$ ). Blocks are interleaved by max-pooling layers.
- ▶ The spatial resolution is progressively reduced, which allows to use more neurons in the top layers.

# The ResNet Architecture



- ▶ Deeper than VGG-16 (ResNets have between 34 and a hundred of layers depending on the variants).
- ▶ Includes skip-connections (identity) every two or three layers in order to make prediction/training more stable.

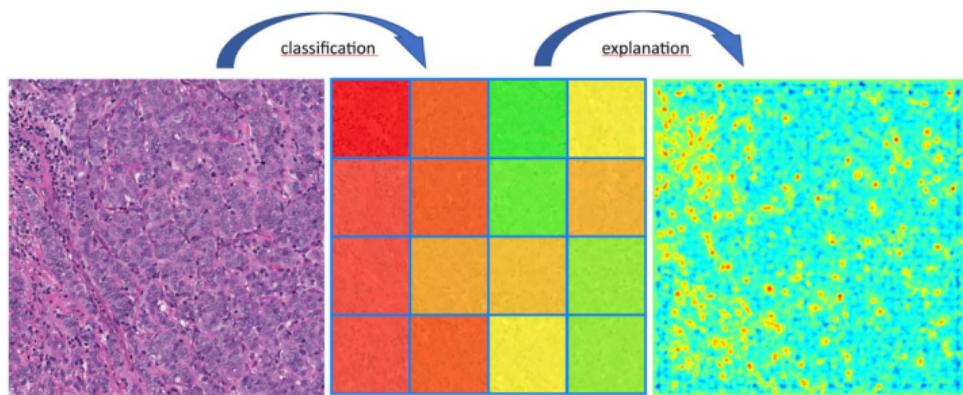
# Application of CNNs: ImageNet Classification



Source: Krizhevsky et al. (2012) ImageNet Classification with Deep Convolutional Neural Networks

- ▶ Ground-truth prediction very often in the top-5 predicted classes (among 1000 predicted classes!).
- ▶ Errors are often similar to errors a human could do (e.g. similar class, or co-located pattern of another class on the same image).

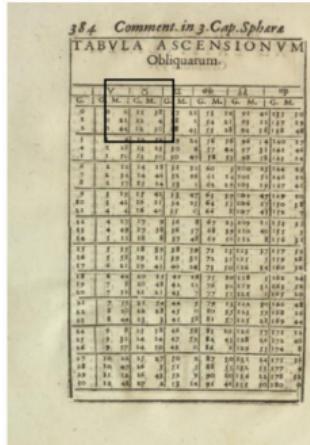
# Application of CNNs: Histopathological Images



Klauschen et al. (2018) Scoring of tumor-infiltrating lymphocytes: from visual estimation to machine learning

- ▶ Histopathological images can be very large ( $\sim$  gigapixel). Convolutional neural networks are trained and applied on small patches.
- ▶ Patch-level predictions can be sharpened spatially by using pixel-wise explanation techniques such as LRP (cf. Lecture 12).

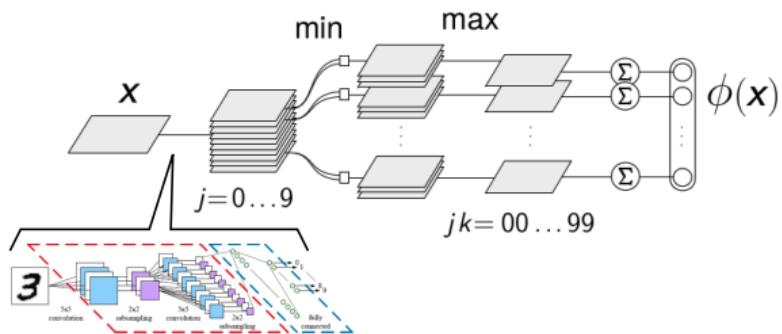
# Specialized Architectures for Table Images



- ▶ Predicting similarity between scanned books content (e.g. from the early-modern era), is a recent direction for making historical inferences.
- ▶ The type of data we would like to predict is very specialized, and standard convnet architectures for image recognition do not work well at predicting this data.
- ▶ We need ad-hoc neural network architectures that take into consideration what types of features are predictive of table similarity, and also that make an efficient usage of data and labels.

# The ‘Bigram Network’ [1]

V		8	
G.	M.	G.	M.
0	0	11	
0	22	12	
0	44	12	



- ▶ Compound architecture composed of a digit recognizer followed by soft-logic operations to extract histogram of digit bigrams.
- ▶ Only the digit recognizer is trained, the rest is hard-coded. This avoids the necessity of having full-table labels.

Part 2

## **Neural Networks for Text**

# Structured Neural Networks for Text

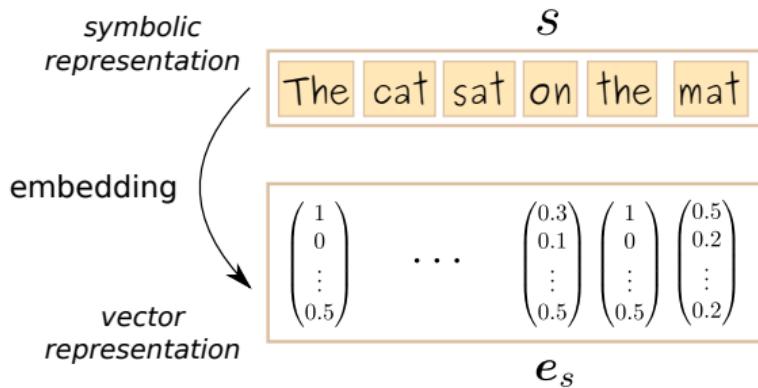
## Image classification

- ▶ Images are arrays of pixels.
- ▶ Pixels  $p$  of an image are given by their RGB components (i.e.  $x_p \in [0, 1]^3$ ). Therefore, they can be readily given as input to the neural network.

## Text classification

- ▶ Texts are collections of words.
- ▶ Words  $w$  are abstract symbols that do not have a numerical representation.  
⇒ They need to be embedded into one such representation so that they can be given as input to a neural network.

# Vector Embedding of a Sentence



# Types of Vector Embedding

1. **One-hot encoding:** Represent the word with indicator functions:

$$e[w] = (1_{\{w='a'\}}, 1_{\{w='able'\}}, 1_{\{w='about'\}}, \dots) \in \mathbb{R}^{\#\text{words}}$$

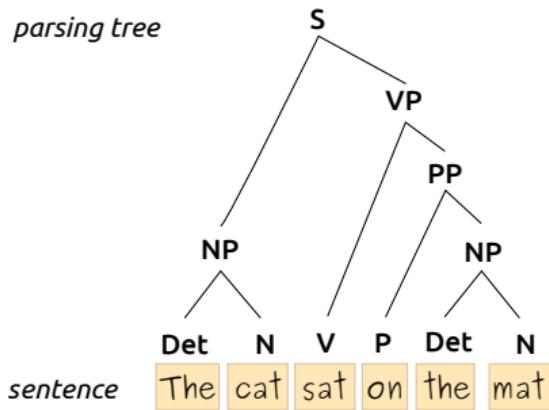
2. **kPCA embedding:** Structured kernel  $k(w, w')$  measures similarity between words (e.g. co-occurrence or lexical similarity). Embedding is obtained by diagonalizing the Gram matrix:  $K = U \Lambda U^\top$ , and defining:

$$e[w] = U_{w,:d} \odot (\lambda_1^{0.5}, \dots, \lambda_d^{0.5}) \in \mathbb{R}^d$$

3. **Learned embedding (e.g. Word2Vec):** Start with a randomly initialized embedding  $e[w] \in \mathbb{R}^d$  and learn the embedding on the supervised task directly or on some related unsupervised task:

$$e[w] \leftarrow e[w] - \gamma \cdot \partial \mathcal{E} / \partial e[w].$$

# Recursive Neural Networks [5]

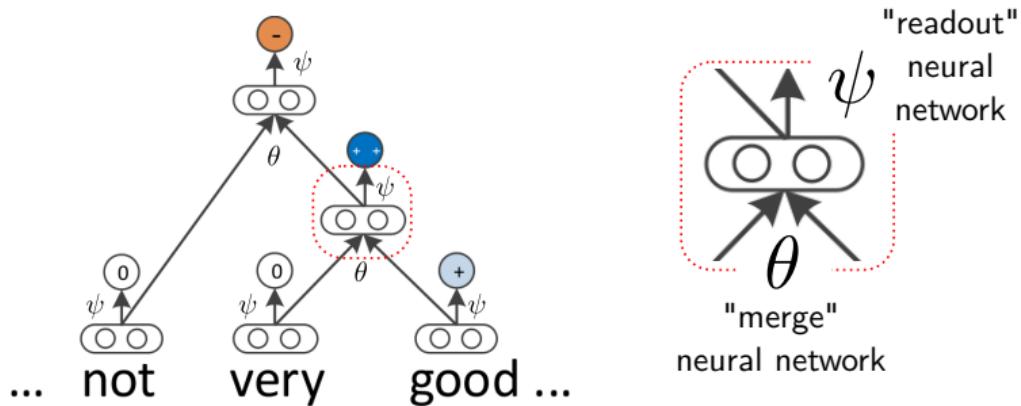


Ideas:

- ▶ Make use of the parsing tree of a sentence to produce a better text classification model.
- ▶ Build the representation by recursively applying a neural network from the leaves of the tree up to the root.

# RecNNs for Sentiment Analysis

The recursive neural network (RecNN) consists of two subnetworks: "merge" and "readout". Both are used at multiple locations in the parse tree.

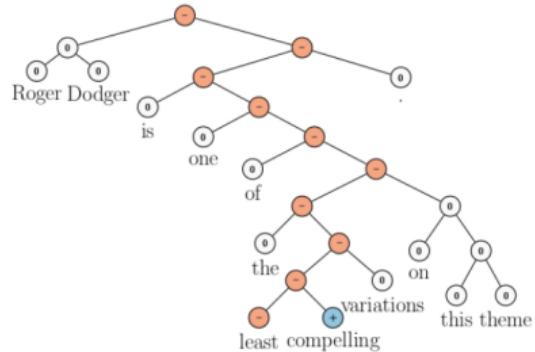
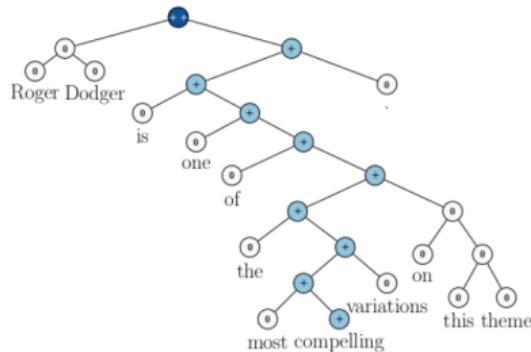


Based on Socher et al. (2013) Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

It can be applied to any sentence that comes with a binary parse tree.

# RecNNs for Sentiment Analysis

Applying the readout function on each node allows to visualize how the sentiment builds up in the recursive neural network.



Source: Socher et al. (2013) Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

# Other Architectures for Text

## Recurrent Neural Networks (LSTMs):

- ▶ Special neural networks that process the text as a sequence.
- ▶ They receive text as input (sequences of word vectors) and can also generate text (sequence of word vectors).
- ▶ They found application e.g. for text translation.
- ▶ They do not require a parse tree associated to the sentence.

## Transformers:

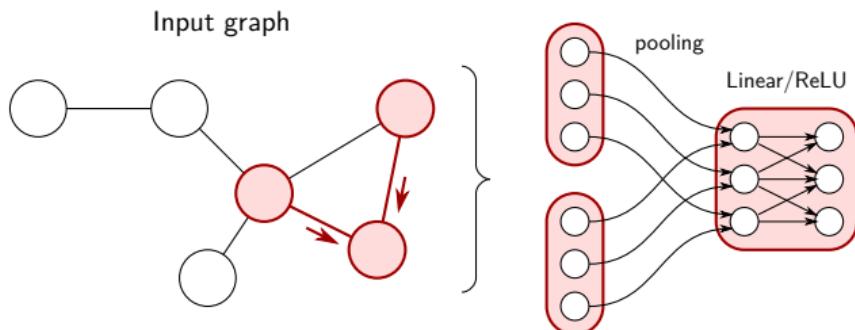
- ▶ Special neural networks that work with input sequences and facilitate the modeling of long-range dependencies in the sequence (via a mechanism called attention).
- ▶ Transformer architectures have become state-of-the-art for a broad range of text processing tasks. They are the basis of large language models such as BERT, GPT-3, etc.

Part 3

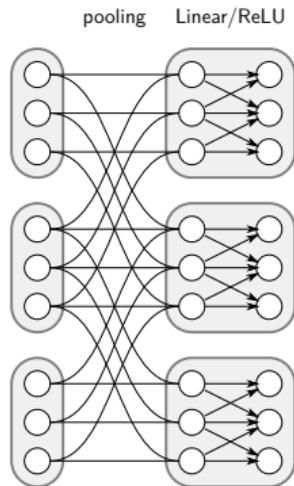
## **Graph Neural Networks**

# Graph Neural Networks

- ▶ Graph neural networks (GNNs) are networks that are specialized for classifying graphs. Graphs are general data representation, that include pixel lattices, trees, sequences as special cases.
- ▶ GNNs receive a graph as input and implement at each layer a propagation step along its edges (via a pooling function).



# Graph Neural Networks



- ▶ Consider a graph of  $N$  nodes. We denote the representation of the graph at layer  $l$  as  $\mathbf{H}_l = (\mathbf{h}_1^{(l)}, \dots, \mathbf{h}_N^{(l)})$  a matrix of dimensions  $N \times d$ .
  - ▶ Define  $\mathbf{\Lambda}$  of size  $N \times N$  a connectivity matrix, e.g. a normalized version of the graph adjacency matrix.
  - ▶ Define  $\mathbf{W}_l$  a matrix of size  $d \times d'$ , which we use to map nodes representations from one layer to another.
- The graph convolutional network [3] defining the following layer-wise mapping:

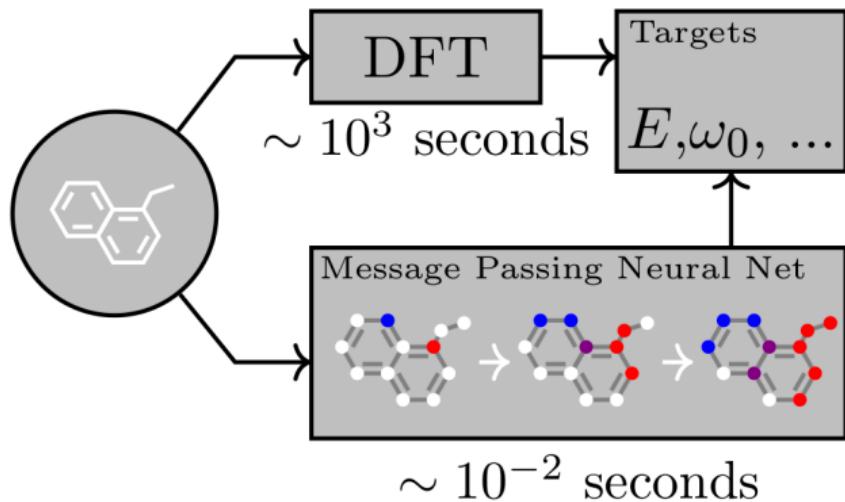
$$\mathbf{A}_l = \mathbf{\Lambda} \mathbf{H}_l \quad \text{(pooling)}$$

$$\mathbf{H}_{l+1} = \rho(\mathbf{A}_l \mathbf{W}_l) \quad \text{(linear/ReLU)}$$

where  $\rho$  is a ReLU function applied element-wise.

# Applications of GNNs

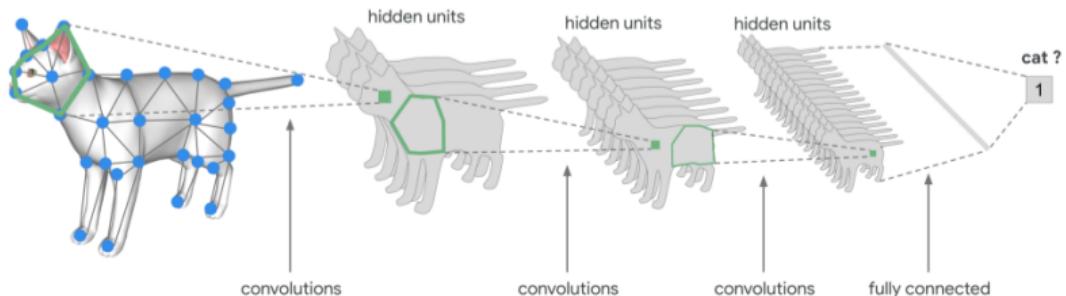
Predicting atomization energy to bypass expensive DFT simulations:



Source: Gilmer et al. (2017) Neural Message Passing for Quantum Chemistry

# Applications of GNNs

Recognizing objects from meshes:

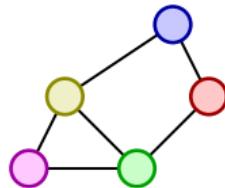


Source: <https://colab.research.google.com>

# Relation between GNNs and Recursive Nets

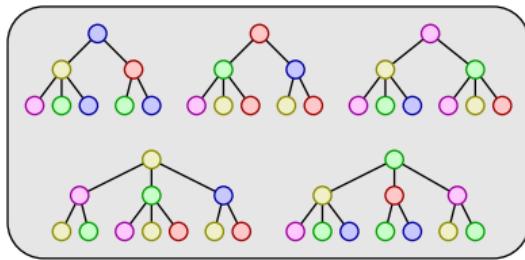
Graph neural networks models can be expanded as recursive neural networks applying to trees. Each tree represents a search of depth  $L$  into the graph starting from a given node, and where  $L$  is the number of layers of the graph neural network.

graph neural network



=

collection of recursive networks



# Relation between GNNs and Convolution Layers

## Observation:

- ▶ The convolution layer can be seen as a special case of a graph neural network where the graph is a two-dimensional lattice.

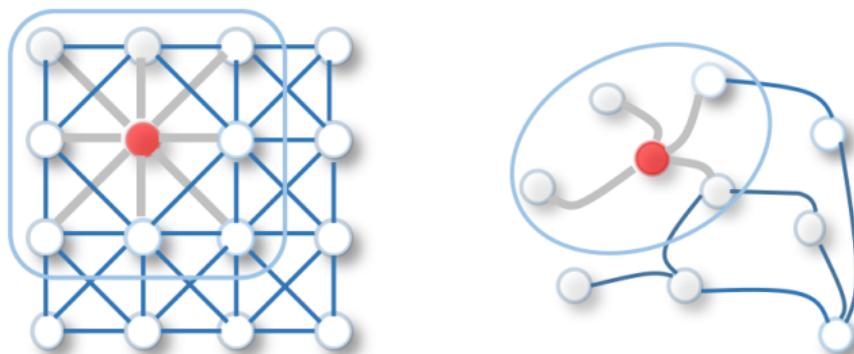


Image source: Wu et al. 2019, A Comprehensive Survey on Graph Neural Networks

# **Summary**

# Summary

- ▶ Structured networks are a generalization of standard neural networks that allows to handle various real-world data (e.g. images, text, graphs). In terms of applications, an analogy can be made between structured neural networks and structured kernels.
- ▶ Convolutional neural networks are a widely used approach for predicting image data. They operate by progressively extracting more complex and useful features while at the same time filtering out exact pixel-wise information.
- ▶ Recursive neural networks enable the processing of data (e.g. text) by following the underlying graph structure of that data (e.g. its parse tree, encoding the grammatical structure).
- ▶ Graph neural networks generalize the application of neural networks to any graph-structured input (including trees and pixel lattices). Convolutional neural networks and recursive neural networks can be seen as a special case of graph neural networks.

# References

-  O. Eberle, J. Büttner, F. Kräutli, K. Müller, M. Valleriani, and G. Montavon.  
Building and interpreting deep similarity models.  
*CoRR*, abs/2003.05431, 2020.
-  K. Fukushima.  
Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.  
*Biological Cybernetics*, 36(4):193–202, 1980.
-  T. N. Kipf and M. Welling.  
Semi-supervised classification with graph convolutional networks.  
In *ICLR (Poster)*. OpenReview.net, 2017.
-  Y. LeCun.  
Generalization and network design strategies.  
In *Connectionism in Perspective*. Elsevier, 1989.
-  R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts.  
Recursive deep models for semantic compositionality over a sentiment treebank.  
In *EMNLP*, pages 1631–1642. ACL, 2013.