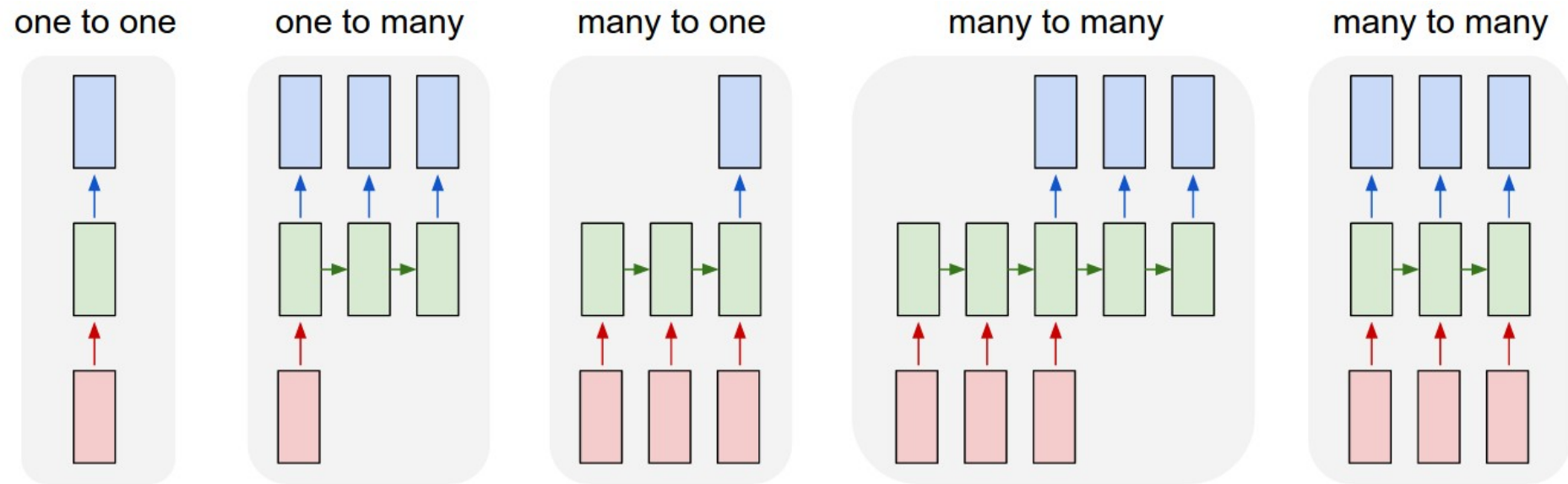
Sequences

Recurrent neural networks (RNNs)

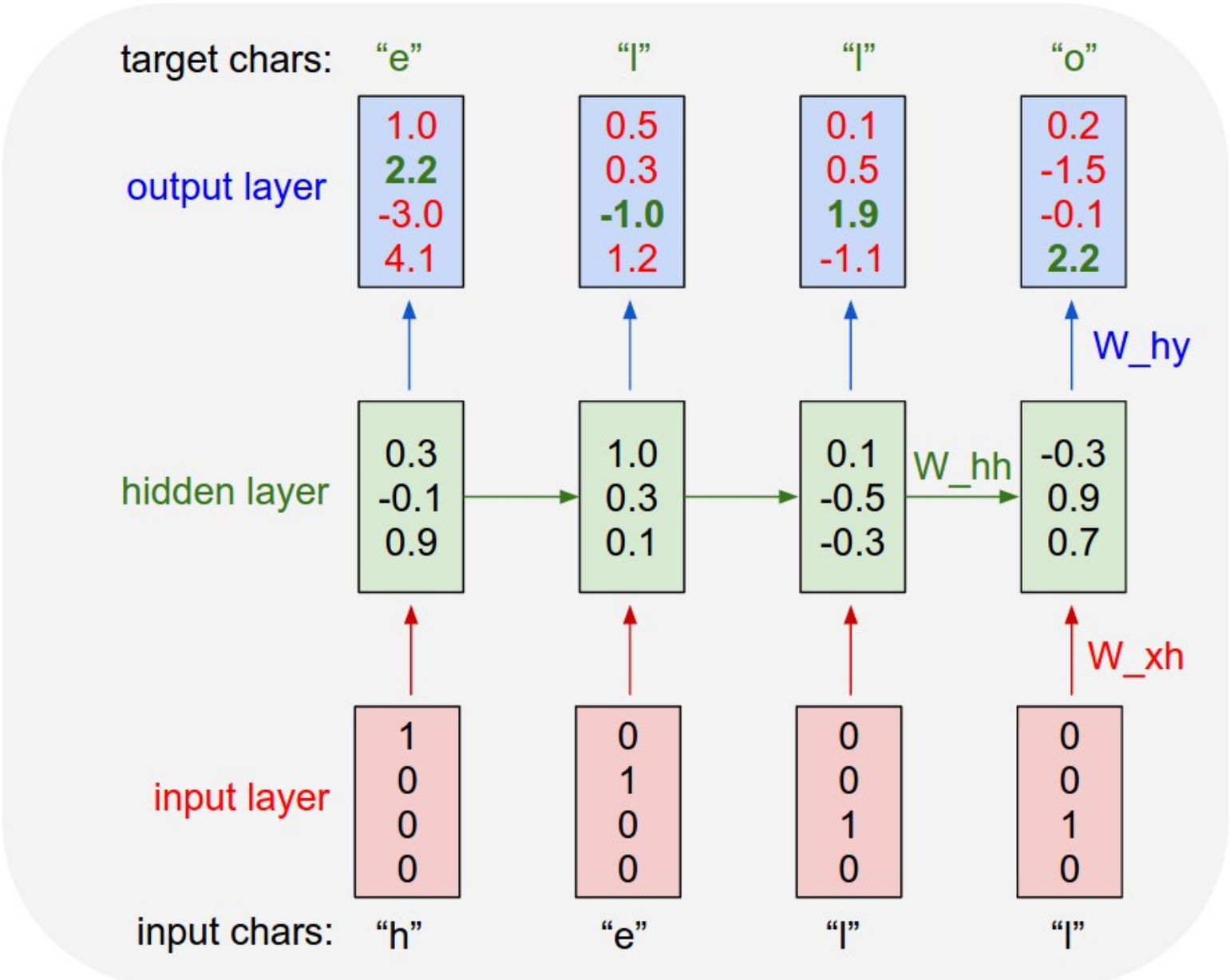
The Unreasonable Effectiveness of Recurrent Neural Networks (Andrej Karpathy 2015)



$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t)$$
$$\mathbf{y}_t = f(\mathbf{W}_{hy}\mathbf{h}_t)$$

Training RNNs (Karpathy 2015)

Training on one word “hello”. (In dropout,remove same nodes in seq.)



Problem of long-term dependencies

How to handle vanishing gradient problem:

*The clouds are in the **sky***

versus

*I grew up in France . . . I speak fluent **French***

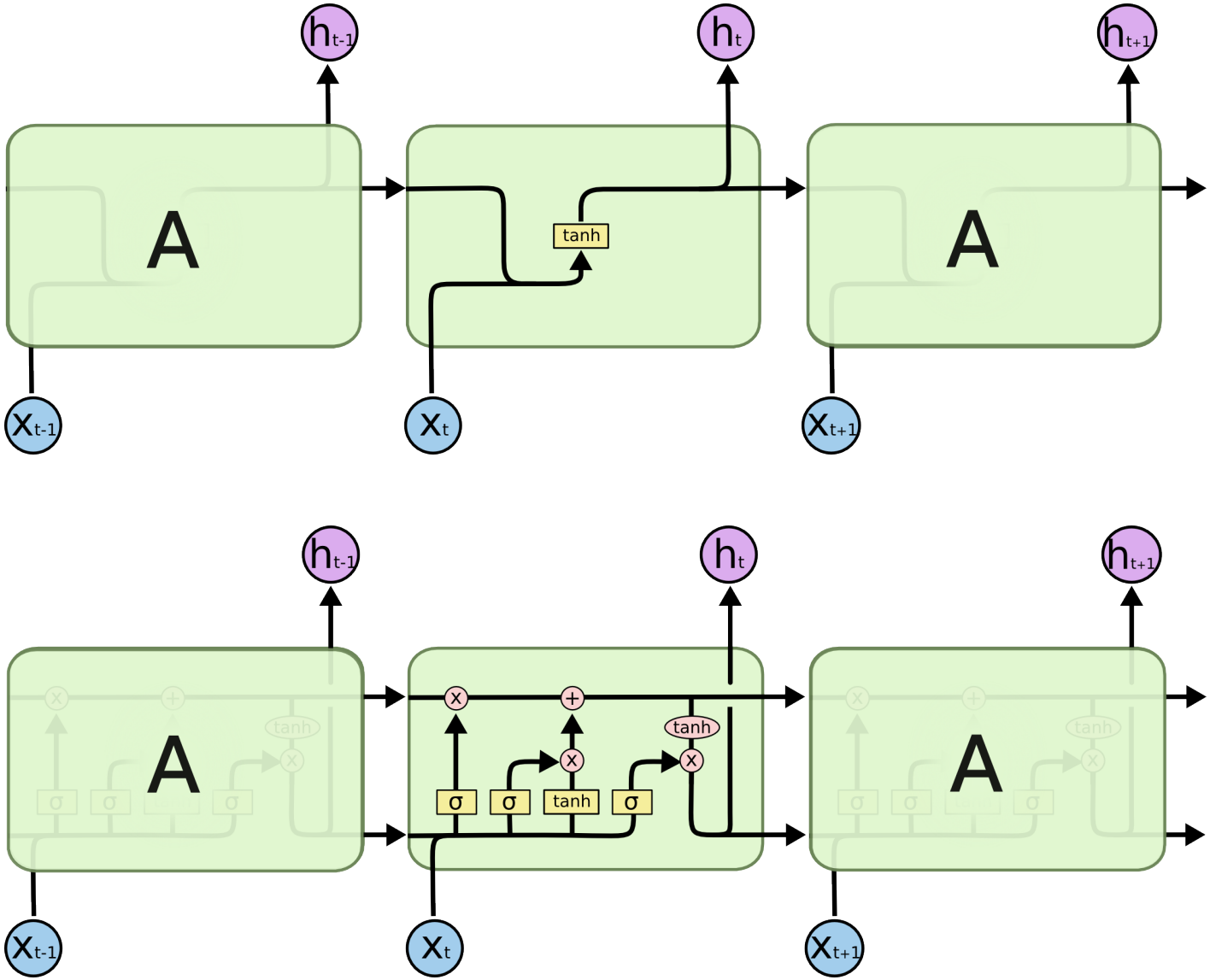
How to keep the relevant memory alive?

Ack: Christopher Olah's blog
`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`

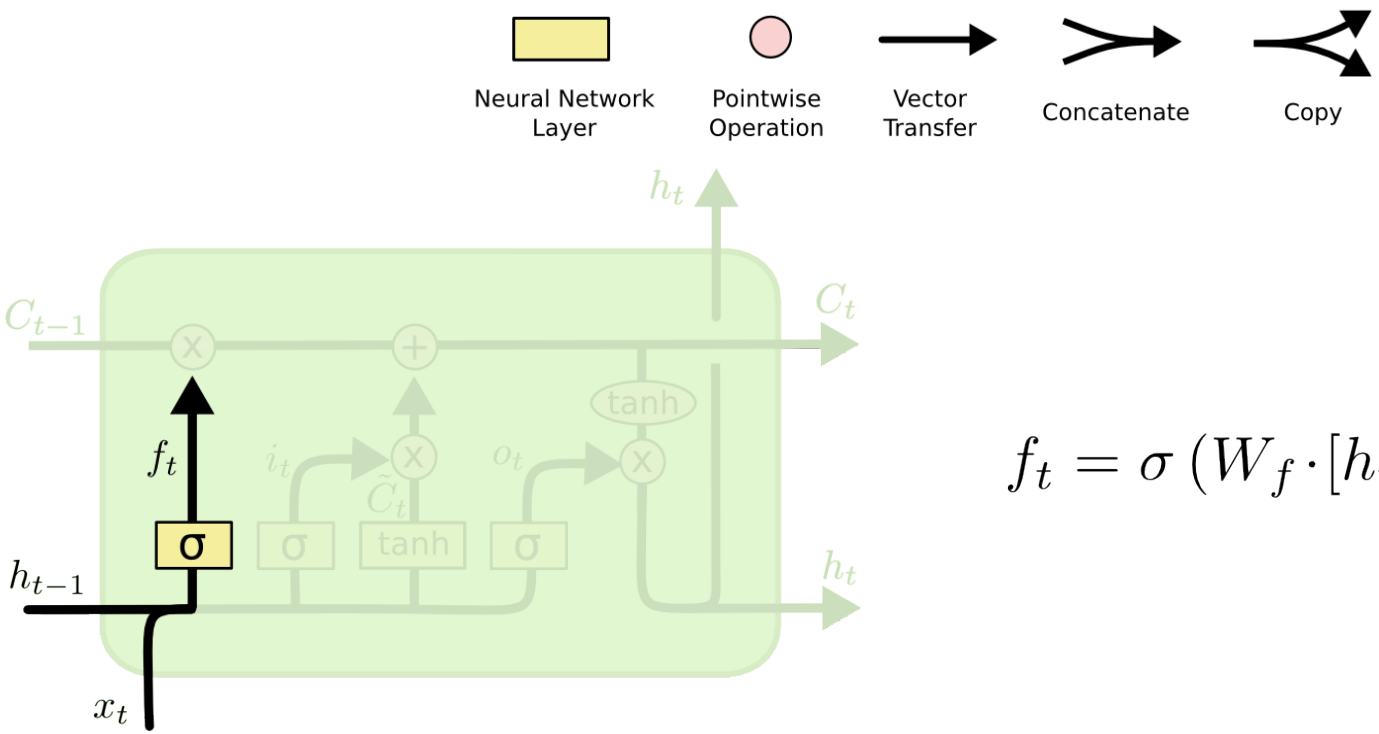
Long short-term memories (LSTM)

- Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780
- Introduce an extra "carry" signal that can propagate a long way through time, and can potentially be modified along the way.
- Can be trained just as with regular units, but more complex.
- Quite complex (see next slide); simpler units (Gated Recurrent Units; GRU) available that are cheaper.
- Key distinction between $\sigma(\cdot)$ and $\tanh(\cdot)$ units.

Comparison of simpleRNN and LSTM (C. Olah)

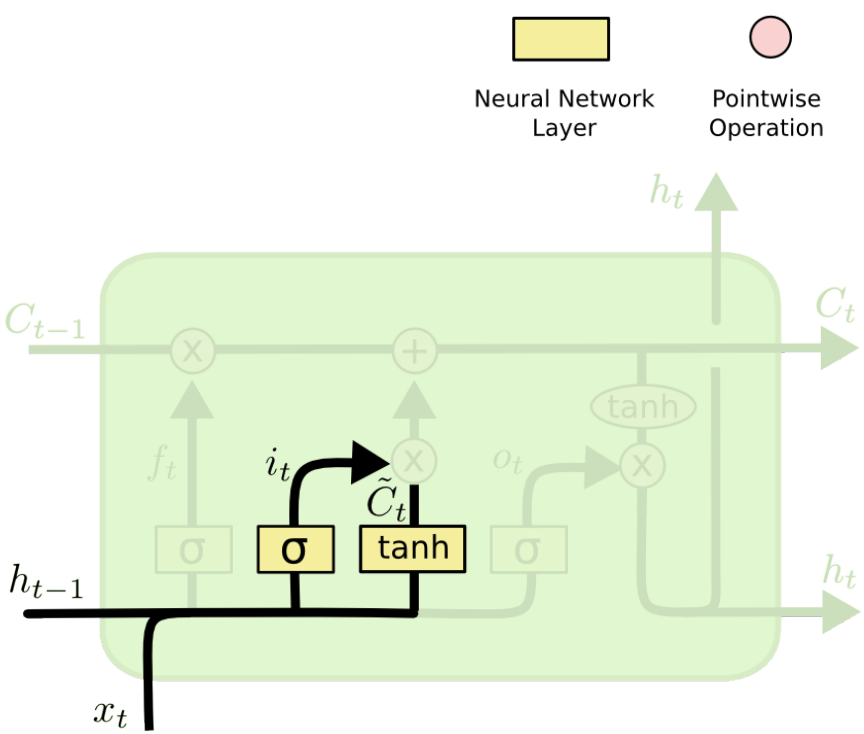


Gating layers: f - forget



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

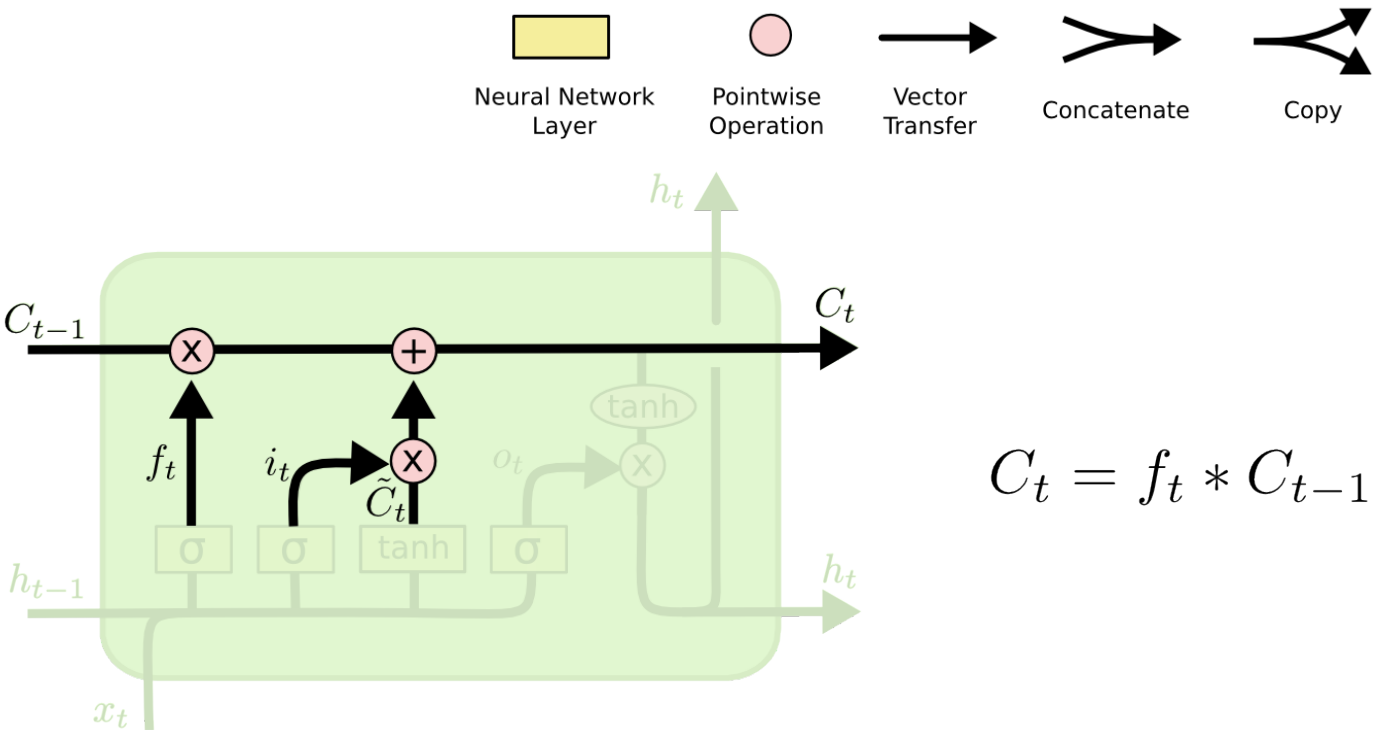
Gating layers: i - input and candidates $\tilde{C}$



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

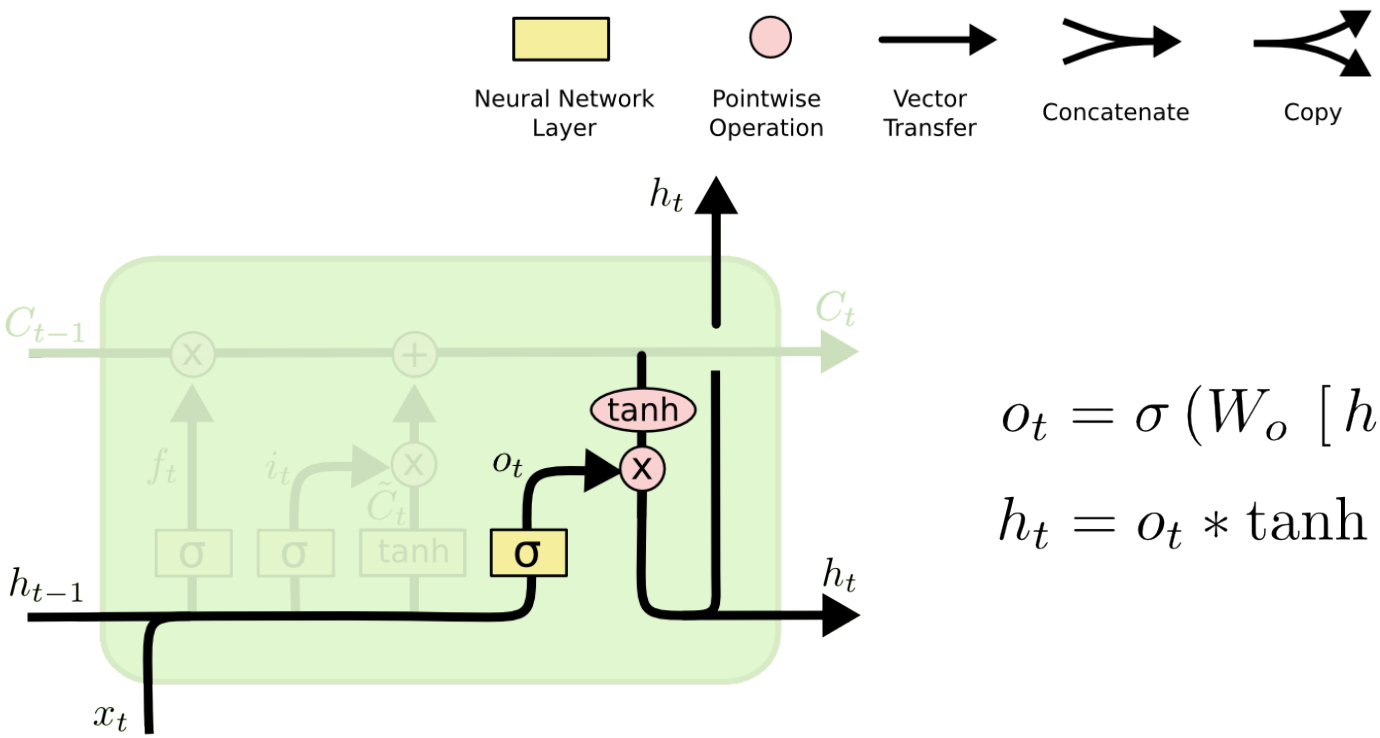
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Update carry signal



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Training framework for character-level RNNs

1. 1 of N coding of input and output characters
2. Hidden layers with state
3. Output layers can be softmax activations. Temperature param influences looseness of speech.
4. Use large corpus of data. These are big networks.

Synthetic Shakespeare (Karpathy 2015)

4.4Mb works of Shakespeare –> 3 layer RNN with 512 hidden nodes in each layer.

> *PANDARUS: Alas, I think he shall be come approached and the day When little srain would be attain'd into being never fed, And who is but a chain and subjects of his death, I should not sleep.*
> *Second Senator: They are away this miseries, produced upon my soul, Breaking and strongly should be buried, when I perish The earth and thoughts of many states.*
> *DUKE VINCENTIO: Well, your wit is in the care of side and that.*
> *Second Lord: They would be ruled after this chamber, and my fair nues begun out of the fact, to be conveyed, Whose noble souls I'll have the heart of the wars.*
> *Clown: Come, sir, I will make did behold your worship.*
> *VIOLA: I'll drink it.*

Synthetic LaTeX (Karpathy 2015)

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

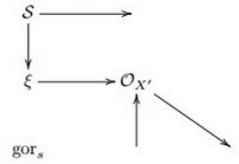
be a morphism of algebraic spaces over S and Y.

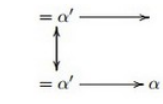
Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram





$\mathrm{Spec}(K_\psi)$ Mor_{Sets} $\mathrm{d}(\mathcal{O}_{X_{X/k}}, \mathcal{G})$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{n}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

Encoding words: sparse vs dense

- One HOT encoding is simple but requires index of words.
- Hashing function can be used to dynamically create index given word. Care needed re: collisions.
- Resulting vectors are large, binary and sparse.
- How about smaller, real-valued dense vectors?
- How to ensure that for two words, their distance in feature space is proportional to semantic distance?
- Or that there are meaningful dimensions to the feature space (cats vs dogs)?

Word2vec: approach

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient Estimation of Word Representations in Vector Space.

Use a net to solve one of two related problems:

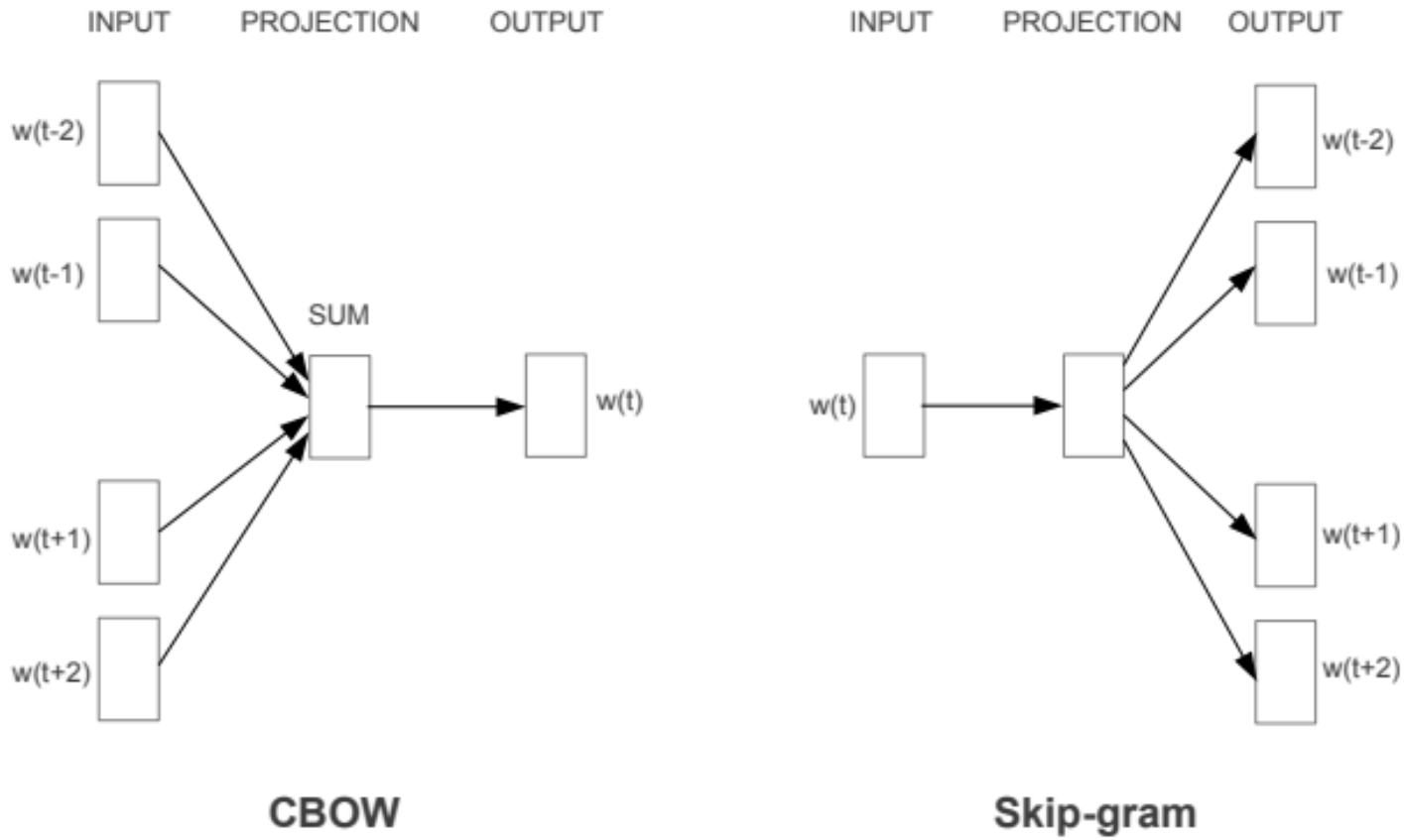


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

What does this feature space look like?

Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed Representations of Words and Phrases and their Compositionality.

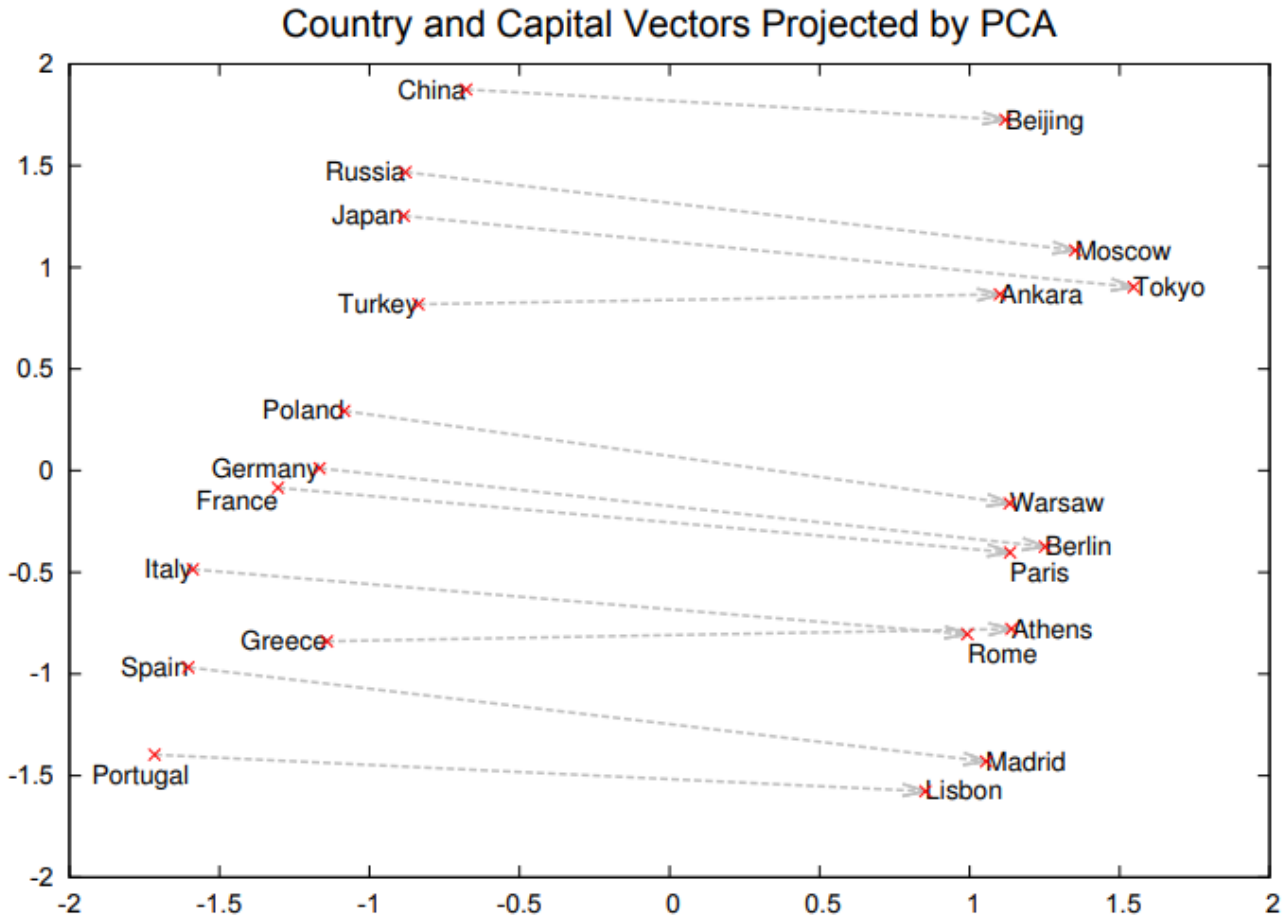


Figure 2: Two-dimensional PCA projection of the 1000-dimensional Skip-gram vectors of countries and their capital cities. The figure illustrates ability of the model to automatically organize concepts and learn implicitly the relationships between them, as during the training we did not provide any supervised information about what a capital city means.

Word2vec: example learned relationships

X = vector(“Paris”) - vector(“France”) + vector(“Italy”). Find word closest to X.

Table 8: *Examples of the word pair relationships, using the best word vectors from Table 4 (Skip-gram model trained on 783M words with 300 dimensionality).*

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Word2vec: technical issues

These networks are large. Corpus of billions of words (from google news). With 10,000 words, input to 300 hidden neurons = 3 million weights.

1. Word pairs: `New York` phrase tokenized into word `New_York`.
2. Subsampling frequent words, e.g. “the”. Probability of keeping a word inversely related to its frequency in database.
3. Negative sampling. Update output weights just for correct word, and randomly chosen 5 incorrect words, rather than all neurons.

See also Stanford’s GloVe `https://nlp.stanford.edu/projects/glove/`
`http://mccormickml.com/2017/01/11/word2vec-tutorial-part-2-negative-sampling/`

Show and tell: a neural image caption generator (Vinyals et al 2015)

Figure 5. A selection of evaluation results, grouped by human rating.

Temporal Convolutional Networks (TCN; Bai et al 2018)

What works in 2d space should work in 1d time. Causal filters and residual networks combined.
Again this is an old idea (Waibel et al 1989), but now have compute power.

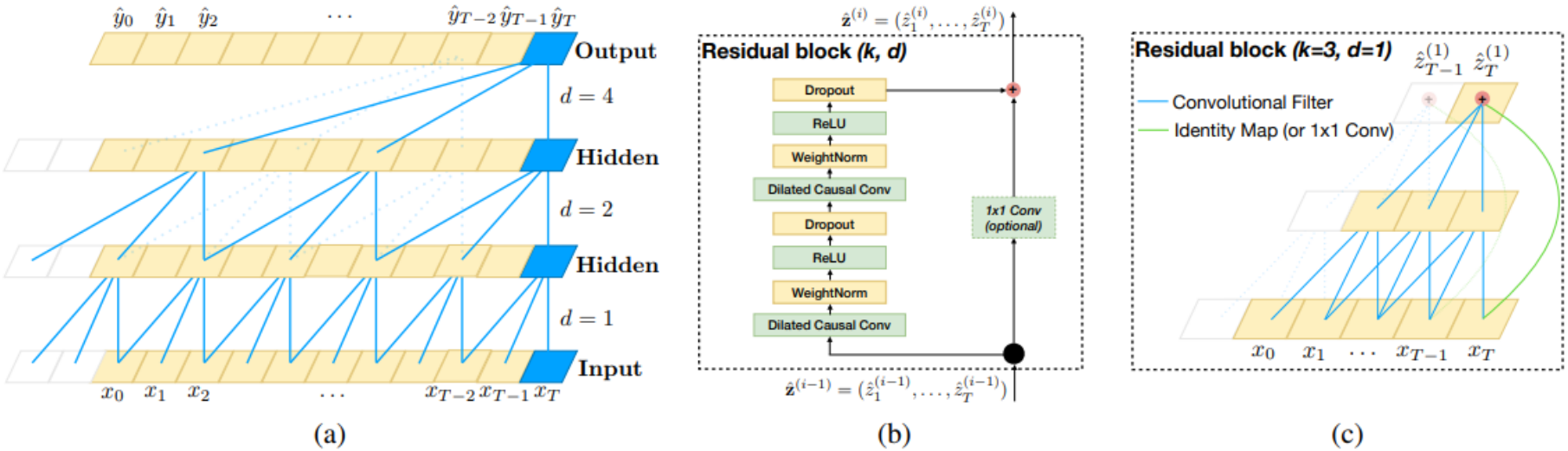


Figure 1. Architectural elements in a TCN. (a) A dilated causal convolution with dilation factors $d = 1, 2, 4$ and filter size $k = 3$. The receptive field is able to cover all values from the input sequence. (b) TCN residual block. An 1x1 convolution is added when residual input and output have different dimensions. (c) An example of residual connection in a TCN. The blue lines are filters in the residual function, and the green lines are identity mappings.

Empirical evaluation (Bai et al 2018)

Table 1. Evaluation of TCNs and recurrent architectures on synthetic stress tests, polyphonic music modeling, character-level language modeling, and word-level language modeling. The generic TCN architecture outperforms canonical recurrent networks across a comprehensive suite of tasks and datasets. Current state-of-the-art results are listed in the supplement. ^{h} means that higher is better. $^{\ell}$ means that lower is better.

Sequence Modeling Task	Model Size ($\approx$)	Models			
		LSTM	GRU	RNN	**TCN**
Seq. MNIST (accuracy^{h})	70K	87.2	96.2	21.5	**99.0**
Permuted MNIST (accuracy)	70K	85.7	87.3	25.3	**97.2**
Adding problem T=600 (loss$^{\ell}$)	70K	0.164	**5.3e-5**	0.177	**5.8e-5**
Copy memory T=1000 (loss)	16K	0.0204	0.0197	0.0202	**3.5e-5**
Music JSB Chorales (loss)	300K	8.45	8.43	8.91	**8.10**
Music Nottingham (loss)	1M	3.29	3.46	4.05	**3.07**
Word-level PTB (perplexity$^{\ell}$)	13M	**78.93**	92.48	114.50	88.68
Word-level Wiki-103 (perplexity)	-	48.4	-	-	**45.19**
Word-level LAMBADA (perplexity)	-	4186	-	14725	**1279**
Char-level PTB (bpc$^{\ell}$)	3M	1.36	1.37	1.48	**1.31**
Char-level text8 (bpc)	5M	1.50	1.53	1.69	**1.45**

Summary

1. Recurrent neural networks (RNNs)
2. Long Short-Term Memories (LSTMs)
3. Word2Vec
4. Temporal Convolutional Networks