Recurrent neural networks
Outline

- Examples of sequential prediction tasks
- Common recurrent units
  - Vanilla RNN unit (and how to train it)
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Recurrent network architectures
- Applications in (a bit) more detail
  - Language modeling
  - Sequence classification
  - Image captioning
- Alternative sequence models
Sequential prediction example 1: Sentiment classification

- Goal: classify a text sequence (e.g., restaurant, movie or product review, Tweet) as having positive or negative sentiment

  - “The food was really good”
  - “The vacuum cleaner broke within two weeks”
  - “The movie had slow parts, but overall was worth watching”

- What makes this problem challenging?
- What feature representation or predictor structure can we use for this problem?
Sequential prediction example 1: Sentiment classification

- Recurrent model:
Sequential prediction example 2: Text generation

- Sample from the distribution of a given text corpus (also known as language modeling)
Sequential prediction example 2: Text generation

• Sample from the distribution of a given text corpus (also known as language modeling)
• Can be done one character or one word at a time:
Sequential prediction example 3: Image captioning

A cat sitting on a suitcase on the floor
A cat is sitting on a tree branch
A dog is running in the grass with a frisbee
A white teddy bear sitting in the grass
Two people walking on the beach with surfboards
A tennis player in action on the court
Two giraffes standing in a grassy field
A man riding a dirt bike on a dirt track

Source: J. Johnson
Captions generated using neuraltalk2
Sequential prediction example 3: Image captioning
Example 4: Machine translation

https://translate.google.com/
Example 4: Machine translation

- Multiple input – multiple output (or sequence to sequence) scenario:

```
"Correspondances" "La" "nature"
```

```
"Matches" "Nature" "is"
```
Summary: Input-output scenarios

- **Single - Single**
  - Feedforward Network

- **Multiple - Single**
  - Sequence Classification

- **Single - Multiple**
  - Sequence generation, captioning

- **Multiple - Multiple**
  - Sequence generation, captioning

- **Multiple - Multiple**
  - Translation
Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
Recurrent unit
Recurrent unit

**Recurrence:**

\[ h_t = f_W(x_t, h_{t-1}) \]

- **new state**
- **function of** \( W \)
- **input at time** \( t \)
- **old state**

- **Output at time** \( t \)
  - \( y_t \)
- **Hidden representation at time** \( t \)
  - \( h_t \)
- **Input at time** \( t \)
  - \( x_t \)

- **Hidden layer**
- **Classifier**
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) \]
\[ = \tanh W \left( \begin{array}{c} x_t \\ h_{t-1} \end{array} \right) \]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) \]
\[ = \tanh W \left( \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \]

\[ \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \]
\[ = 2\sigma(2a) - 1 \]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) \]

\[ = \tanh W \left( \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \right) \]

\[ \frac{d}{da} \tanh(a) = 1 - \tanh^2(a) \]
Vanilla RNN cell

\[ h_t = f_W(x_t, h_{t-1}) \]
\[ = \tanh \left( W_x x_t + W_h h_{t-1} \right) \]
RNN forward pass

\[ e_t = -\log(y_t(GT_t)) \]
\[ y_t = \text{softmax}(W_y h_t) \]
\[ h_t = \text{tanh} \left( W \left( \begin{array}{c} x_t \\ h_{t-1} \end{array} \right) \right) \]

----- shared weights
RNN forward pass: Computation graph

\[
e_t = -\log(y_t(GT_t))
\]

\[
y_t = \text{softmax}(W_y h_t)
\]

\[
h_t = \tanh(W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix})
\]
Training: Backpropagation through time (BPTT)

- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input.
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights.
Backpropagation through time

Forward through entire sequence to compute loss, then backward to compute gradient

Source: J. Johnson
Backpropagation through time

Problem: Takes a lot of memory for long sequences!

Source: J. Johnson
Training: Backpropagation through time (BPTT)

• The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
• The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights
• In practice, truncated BPTT is used: run the RNN forward \( k_1 \) time steps, propagate backward for \( k_2 \) time steps

https://machinelearningmastery.com/gentle-introduction-backpropagation-time/
Truncated backpropagation through time

Run forward and backward through chunks of the sequence instead of whole sequence

Source: J. Johnson
Truncated backpropagation through time

Carry hidden states forward in time farther, but only backpropagate for some smaller number of steps

Source: J. Johnson
Truncated backpropagation through time

Source: J. Johnson
RNN backward pass

Parameter update

\[
\frac{\partial e}{\partial W} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) h_{t-1}^T
\]

\[
\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot (1 - \tanh^2(W_x x_t + W_h h_{t-1})) x_t^T
\]

\[
\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}
\]

Error from predictions at future steps

\[
h_t = \tanh(W_x x_t + W_h h_{t-1})
\]
Vanishing and exploding gradients

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T (1 - \tanh^2(W_x x_t + W_h h_{t-1})) \odot \frac{\partial e}{\partial h_t}$$

Computing gradient for $h_0$ involves many multiplications by $W_h^T$ and rescalings between 0 and 1.

Gradients will vanish if largest singular value of $W_h$ is less than 1 and explode if it’s greater than 1.
Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit (and how to train it)
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
Long short-term memory (LSTM)

• Add a memory cell that is not subject to matrix multiplication or squishing, thereby avoiding gradient decay

The LSTM cell

\[ g_t = \tanh(W_g x_t) \]

\[ h_t = \text{some function of } g_t, x_t, h_{t-1} \]
The LSTM cell

\[ g_t = \tanh(W_g x_t) \]

\[ c_t = c_{t-1} + g_t \]

\[ h_t = \tanh(c_t) \]
The LSTM cell

\[
g_t = \tanh(W_g x_t + W_{h_t} h_{t-1})
\]
The LSTM cell

\[
\begin{align*}
    i_t &= \sigma \left( W_i \left( x_t, h_{t-1} \right) + b_i \right) \\
    g_t &= \tanh \left( W_g \left( x_t, h_{t-1} \right) \right) \\
    c_t &= c_{t-1} + i_t \odot g_t \\
    h_t &= \sigma \left( c_t, h_{t-1} \right)
\end{align*}
\]
The LSTM cell

The diagram shows the structure of an LSTM (Long Short-Term Memory) cell, which is a type of recurrent neural network (RNN) cell designed to overcome the vanishing gradient problem. The LSTM cell consists of three main gates: the input gate, the forget gate, and the output gate, along with a cell state. The equations and operations for these gates and states are as follows:

- **Input Gate** 
  \[ i_t = \sigma(W_i x_t + b_i) \]
- **Forget Gate** 
  \[ f_t = \sigma(W_f h_{t-1} + b_f) \]
- **Output Gate** 
  \[ o_t = \sigma(W_o h_{t-1} + b_o) \]
- **Cell** 
  \[ c_t = f_t \odot c_{t-1} + i_t \odot g_t \]
- **Cell State** 
  \[ h_t = o_t \odot \tanh(c_t) \]

Where \( i_t, f_t, o_t \) are the gates, \( x_t, h_{t-1} \) are the input and previous hidden state, \( c_t \) is the cell state, and \( \sigma \) is the sigmoid activation function, \( \odot \) is the Hadamard product (element-wise multiplication), and \( \tanh \) is the hyperbolic tangent activation function. The diagram illustrates how these operations flow through the cell, with arrows indicating the direction of information flow.
The LSTM cell

\[
\begin{align*}
\text{Input Gate} & : i_t = \sigma \left( W_i \left( x_t, h_{t-1} \right) + b_i \right) \\
\text{Forget Gate} & : f_t = \sigma \left( W_f \left( x_t, h_{t-1} \right) + b_f \right) \\
\text{Cell} & : g_t = \tanh \left( W_g \left( x_t, h_{t-1} \right) \right) \\
\text{Output Gate} & : o_t = \sigma \left( W_o \left( h_{t-1} \right) + b_o \right) \\
\text{Cell State} & : c_t = f_t \odot c_{t-1} + i_t \odot g_t \\
\text{Hidden State} & : h_t = o_t \odot \tanh c_t
\end{align*}
\]
LSTM forward pass summary

\[
\begin{align*}
\mathbf{g}_t &= \begin{pmatrix} \tanh \mathbf{z}_t \\ \sigma \mathbf{r}_t \end{pmatrix} \begin{pmatrix} \mathbf{W}_g \\ \mathbf{W}_i \\ \mathbf{W}_f \\ \mathbf{W}_o \end{pmatrix} \mathbf{x}_t \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh \mathbf{c}_t
\end{align*}
\]
LSTM backward pass

Gradient flow from $c_t$ to $c_{t-1}$ only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh.

For complete details: Illustrated LSTM Forward and Backward Pass
LSTM variant: Gated recurrent unit (GRU)

- Get rid of separate cell state
- Merge “forget” and “output” gates into “update” gate

\[
\begin{align*}
    r_t &= \sigma \left( W_r \left( x_t \right) + b_t \right) \\
    h'_t &= \tanh \left( W \left( r_t \odot h_{t-1} \right) \right) \\
    z_t &= \sigma \left( W_z \left( x_t \right) + b_z \right) \\
    h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot h'_t
\end{align*}
\]

Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit (and how to train it)
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
Recall: Input-output scenarios

Multiple - Single
Multiple - Multiple
Single - Multiple
Multiple - Multiple

Sequence generation, captioning
Sequence generation, captioning
Sequence generation, captioning
Translation

Recall: Input-output scenarios
RNN architectures

• Most general configuration:
Multi-layer RNNs

• We can of course design RNNs with multiple hidden layers

\[ \begin{align*}
\mathbf{x}_1 & \rightarrow \mathbf{y}_1 \\
\mathbf{x}_2 & \rightarrow \mathbf{y}_2 \\
\mathbf{x}_3 & \rightarrow \mathbf{y}_3 \\
\mathbf{x}_4 & \rightarrow \mathbf{y}_4 \\
\mathbf{x}_5 & \rightarrow \mathbf{y}_5 \\
\mathbf{x}_6 & \rightarrow \mathbf{y}_6
\end{align*} \]

• Anything goes: skip connections across layers, across time, …
Multi-layer RNNs

• We can of course design RNNs with multiple hidden layers

• Anything goes: skip connections across layers, across time, …
Multi-layer RNNs

• We can of course design RNNs with multiple hidden layers

\[ y_1 y_2 y_3 y_4 y_5 y_6 \]

\[ x_1 x_2 x_3 x_4 x_5 x_6 \]

• Anything goes: skip connections across layers, across time, …
Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction (common in speech recognition)
Google Neural Machine Translation (GNMT)

Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
• Applications in (a bit) more detail
  • Language modeling
  • Sequence classification
  • Image captioning
  • Machine translation (later)
Language modeling: Character RNN

Output symbol $y_i$

Output layer (linear + softmax)

Hidden state $h_i$

One-hot encoding $x_i$

Input symbol

$p(y_1, y_2, \ldots, y_n)$

$= \prod_{i=1}^{n} p(y_i | y_1, \ldots, y_{i-1})$

$\approx \prod_{i=1}^{n} P_W(y_i | h_i)$

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Language modeling: Character RNN

100th iteration

tyntd-iafhatawiaohrdemot lytdws e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e plia tk8rgd t 0 idoe ns,smtt h ne etie h,hrgrtrs nigiteke,aoaenns lng

300th iteration

"Tmont thithey" fomesscerliund Keushey. Thom here shelu, amnerenith ol sivh I lalterthend Bleipile shuw y fil on aseterlome coaniognc Phe lism thond hon at. MeGImotion in ther thize."

700th iteration

After fall unsuch that the hall for Prince Velzonski's that me of her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort how, and Gogition is so overelical and ofter.

2000th iteration

"Why do what that day," replied Natasha, and wishing to himself the fact the princess, Princess Mary was easier, fed in had oftended him. Pierre aking his soul came to the packs and drove up his father-in-law women.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Searching for interpretable hidden units

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Searching for interpretable hidden units

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

line position tracking cell

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Searching for interpretable hidden units

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Searching for interpretable hidden units

Searching for interpretable hidden units

code depth cell
Searching for interpretable hidden units

A. Karpathy, J. Johnson, and L. Fei-Fei, Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016
Sequence classification

“The food is usually not so good”

Word embedding  \( W_e \cdot x_1 \)

One-hot encoding  \( x_1 \)

“The”
Sequence classification

Embed

x_1

x_2

W_e x_1

W_e x_2

h_1

h_2

h_{n-1}

RNN

RNN

RNN

“The”

“food”

“good”
Sequence classification

```
RNN

"The"
```

```
RNN

"food"
```

```
RNN

"good"
```

```
Sequence classification

http://deeplearning.net/tutorial/lstm.html
Sequence classification

\[ h = \text{Sum}(...) \]

Bi-RNN

Embed

"The"

Embed

"food"

Embed

"good"

Linear Classifier
Recurrent models: Outline

- Examples of sequential prediction tasks
- Common recurrent units
  - Vanilla RNN unit
  - Long Short-Term Memory (LSTM)
  - Gated Recurrent Unit (GRU)
- Recurrent network architectures
- Applications in (a bit) more detail
  - Language modeling
  - Sequence classification
  - Image captioning
Image caption generation

Training time

• Maximize likelihood of reference captions

Image caption generation: Test time

- How do we produce a caption given a test image?
  - How about always choosing the highest-likelihood word?
  - Are there any problems with this approach?
Image caption generation: Beam search

- Maintain $k$ (beam width) top-scoring candidate sentences according to sum of per-word log-likelihoods (or some other score)
- At each step, generate all their successors and keep the best $k$
Image caption generation: Beam search

```
The
A
One

The dog is hiding.
A yellow dog is sitting on the.
One cat is sitting.
```

```
Classifier
Classifier
Classifier
Classifier
Classifier

h0  h1  h2  h3  h4  h5

CNN

“START”

[Image of a dog and cat]
```
Image caption generation: Example outputs
How to evaluate image captioning?

Reference sentences (written by human annotators):

- “A dog hides underneath a bed with its face peaking out of the bed skirt”
- “The small white dog is peeking out from under the bed”
- “A dog is peeking its head out from underneath a bed skirt”
- “A dog peeking out from under a bed”
- “A dog that is under a bed on the floor”

Generated sentence:

- “A dog is hiding”
BLEU: Bilingual Evaluation Understudy

- **N-gram precision**: count the number of n-gram matches between candidate and reference translation, divide by total number of n-grams in candidate translation
  - Clip counts by the maximum number of times an n-gram occurs in any reference translation
  - Multiply by *brevity penalty* to penalize short translations

- Most commonly used measure for image captioning and machine translation despite multiple shortcomings

BLEU: Bilingual Evaluation Understudy

Original (French): J’ai mangé la pomme.

Reference translation: I ate the apple.

Based on BLEU, these are all “equally bad” output sentences.

I consumed the apple.

I ate an apple.

I ate the potato.

<table>
<thead>
<tr>
<th>Model</th>
<th>CIDEr-D</th>
<th>Meteor</th>
<th>ROUGE-L</th>
<th>BLEU-1</th>
<th>BLEU-2</th>
<th>BLEU-3</th>
<th>BLEU-4</th>
</tr>
</thead>
<tbody>
<tr>
<td>m-RNN (Baidu/ UCLA)</td>
<td>0.886</td>
<td>0.238</td>
<td>0.524</td>
<td>0.72</td>
<td>0.553</td>
<td>0.41</td>
<td>0.302</td>
</tr>
<tr>
<td>m-RNN</td>
<td>0.842</td>
<td>0.250</td>
<td>0.452</td>
<td>0.586</td>
<td>0.446</td>
<td>0.303</td>
<td>0.229</td>
</tr>
<tr>
<td>MSR Captivate</td>
<td>0.830</td>
<td>0.257</td>
<td>0.476</td>
<td>0.594</td>
<td>0.459</td>
<td>0.349</td>
<td>0.272</td>
</tr>
<tr>
<td>Google</td>
<td>0.868</td>
<td>0.258</td>
<td>0.488</td>
<td>0.603</td>
<td>0.461</td>
<td>0.349</td>
<td>0.287</td>
</tr>
<tr>
<td>Berkeley LR+</td>
<td>0.791</td>
<td>0.226</td>
<td>0.416</td>
<td>0.535</td>
<td>0.413</td>
<td>0.294</td>
<td>0.225</td>
</tr>
<tr>
<td>Nearest Neighbor</td>
<td>0.618</td>
<td>0.161</td>
<td>0.319</td>
<td>0.417</td>
<td>0.319</td>
<td>0.262</td>
<td>0.202</td>
</tr>
<tr>
<td>MSR[8]</td>
<td>0.893</td>
<td>0.233</td>
<td>0.537</td>
<td>0.704</td>
<td>0.558</td>
<td>0.422</td>
<td>0.309</td>
</tr>
<tr>
<td>Montreal/Toronto[10]</td>
<td>0.823</td>
<td>0.236</td>
<td>0.521</td>
<td>0.687</td>
<td>0.518</td>
<td>0.382</td>
<td>0.281</td>
</tr>
<tr>
<td>PicSOM[13]</td>
<td>0.873</td>
<td>0.241</td>
<td>0.525</td>
<td>0.692</td>
<td>0.521</td>
<td>0.397</td>
<td>0.291</td>
</tr>
<tr>
<td>Tsinghua Bigeye[14]</td>
<td>0.673</td>
<td>0.207</td>
<td>0.49</td>
<td>0.671</td>
<td>0.494</td>
<td>0.35</td>
<td>0.241</td>
</tr>
<tr>
<td>MLBL[7]</td>
<td>0.760</td>
<td>0.219</td>
<td>0.499</td>
<td>0.666</td>
<td>0.498</td>
<td>0.362</td>
<td>0.26</td>
</tr>
<tr>
<td>Human[9]</td>
<td>0.854</td>
<td>0.219</td>
<td>0.484</td>
<td>0.663</td>
<td>0.469</td>
<td>0.321</td>
<td>0.217</td>
</tr>
</tbody>
</table>

### Metrics

- **CIDEr-D**: Consensus-based Image Description Evaluation
- **METEOR**: Meteor Universal: Language Specific Translation Evaluation for Any Target Language
- **ROUGE-L**: ROUGE: A Package for Automatic Evaluation of Summaries
- **BLEU**: BLEU: a Method for Automatic Evaluation of Machine Translation

[http://mscoco.org/dataset/#captions-leaderboard](http://mscoco.org/dataset/#captions-leaderboard)
<table>
<thead>
<tr>
<th>Model</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>0.638</td>
<td>0.675</td>
<td>4.836</td>
<td>3.428</td>
<td>0.352</td>
</tr>
<tr>
<td>Google</td>
<td>0.728</td>
<td>0.413</td>
<td>4.407</td>
<td>3.740</td>
<td>0.233</td>
</tr>
<tr>
<td>MSR</td>
<td>0.973</td>
<td>0.347</td>
<td>4.407</td>
<td>3.740</td>
<td>0.233</td>
</tr>
<tr>
<td>Montreal</td>
<td>0.216</td>
<td>0.255</td>
<td>3.801</td>
<td>2.716</td>
<td>0.196</td>
</tr>
<tr>
<td>MSR-Caption</td>
<td>0.202</td>
<td>0.250</td>
<td>3.965</td>
<td>2.552</td>
<td>0.182</td>
</tr>
<tr>
<td>Berkeley</td>
<td>0.194</td>
<td>0.213</td>
<td>3.079</td>
<td>3.482</td>
<td>0.154</td>
</tr>
<tr>
<td>m-RNN (Baidu/ UCL)</td>
<td>0.190</td>
<td>0.241</td>
<td>3.831</td>
<td>2.548</td>
<td>0.195</td>
</tr>
<tr>
<td>MIL</td>
<td>0.168</td>
<td>0.197</td>
<td>3.349</td>
<td>2.915</td>
<td>0.159</td>
</tr>
<tr>
<td>MLBL</td>
<td>0.167</td>
<td>0.196</td>
<td>3.659</td>
<td>2.420</td>
<td>0.156</td>
</tr>
</tbody>
</table>

**Legend:**
- **M1**: Percentage of captions that are evaluated as better or equal to human caption.
- **M2**: Percentage of captions that pass the Turing Test.
- **M3**: Average correctness of the captions on a scale 1-5 (incorrect - correct).
- **M4**: Average amount of detail of the captions on a scale 1-5 (lack of details - very detailed).
- **M5**: Percentage of captions that are similar to human description.
Outline

• Examples of sequential prediction tasks
• Common recurrent units
  • Vanilla RNN unit
  • Long Short-Term Memory (LSTM)
  • Gated Recurrent Unit (GRU)
• Recurrent network architectures
• Applications in (a bit) more detail
  • Language modeling
  • Sequence classification
  • Image captioning
• Alternative sequence models
Convolutional sequence models

- **Recurrent models:**
  - Treat input as ordered sequence (inherently sequential processing)
  - Build up context using the hidden vector

- **Convolutional models:**
  - Treat input as a grid indexed by time and feature dimension
  - Build up context using multiple layers of convolutions
  - Processing can be parallel at training time, but convolutions must be *causal*
WaveNet

- Goal: generate raw audio
  - Represented as sequence of 16-bit integer values (can be quantized to 256 discrete levels), 16K samples per second
- Applications: text-to-speech, music generation
  - Also works for speech recognition

A. van den Oord et al., WaveNet: A generative model for raw audio, arXiv 2016
WaveNet

- Training time: compute predictions of all timesteps in parallel (conditioned on ground truth)
WaveNet

- Test time: feed each predicted sample back into the model to make prediction at next timestep
WaveNet: Results

• Text-to-speech with different speaker identities:

• Generated sample of classical piano music:

https://deepmind.com/blog/article/wavenet-generative-model-raw-audio
Temporal convolutional networks (TCNs)

- TCNs can be competitive with RNNs for a variety of sequence modeling tasks

---