# Graves 2013, "Generating Sequences with Recurrent Neural Networks"

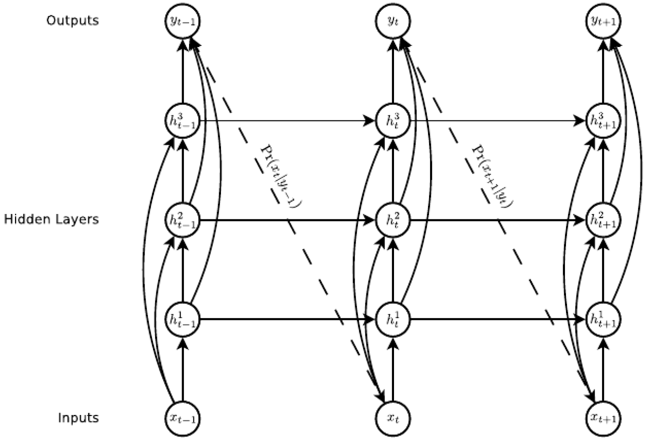
Johannes Bausch and Jack Kamm

14 November 2017

# Outline

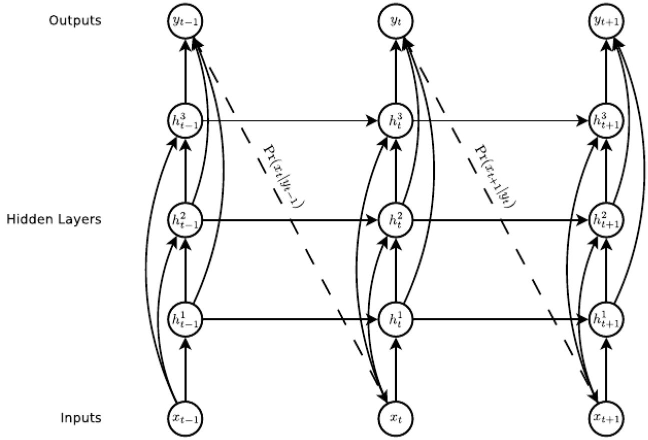
# Table of Contents

# Recurrent Neural Networks




Input $x_t$, hidden layers $h_t^n$, output $y_t$, generative model $\mathbb{P}(x_{t+1} \mid y_t)$

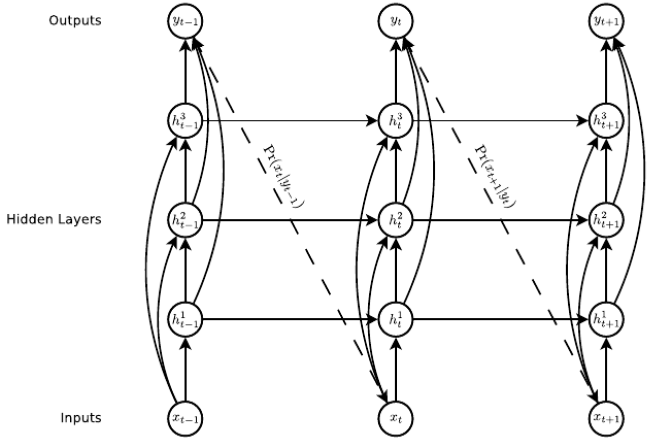
# Recurrent Neural Networks




$h_t^n$ = nonlinear link $\circ$ affine combo of $x_t$, $h_{t-1}^n$, $h_t^{n-1}$

$$h_t^n = \mathcal{H}(W_{ih^n} x_t + W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n)$$

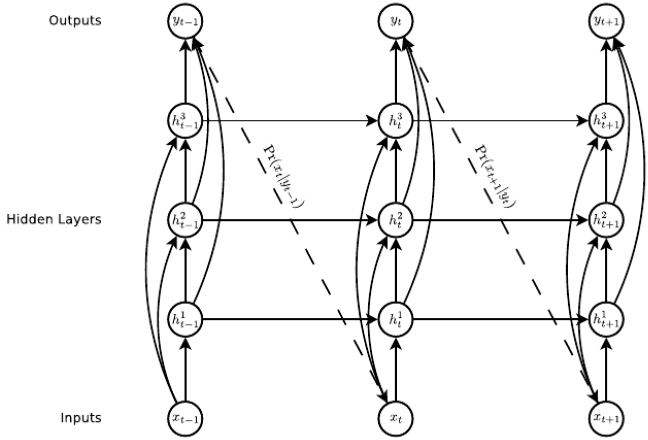
# Recurrent Neural Networks




$y_t$ = nonlinear link $\circ$ affine combo of $h^n_t$

$$y_t = \mathcal{Y}(b_y + \sum_{n=1}^{N} W_{h^n y} h^n_t)$$

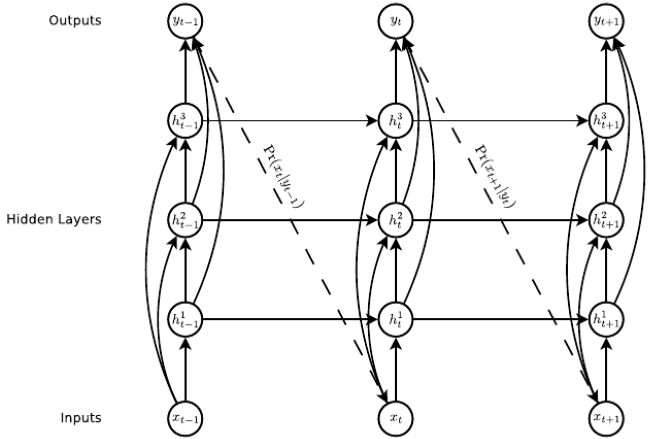
# Recurrent Neural Networks




Train by maximizing likelihood of generative model:

$$\mathbb{P}(\mathbf{x}) = \prod_{t=1}^{T} \mathbb{P}(x_t \mid y_{t-1})$$

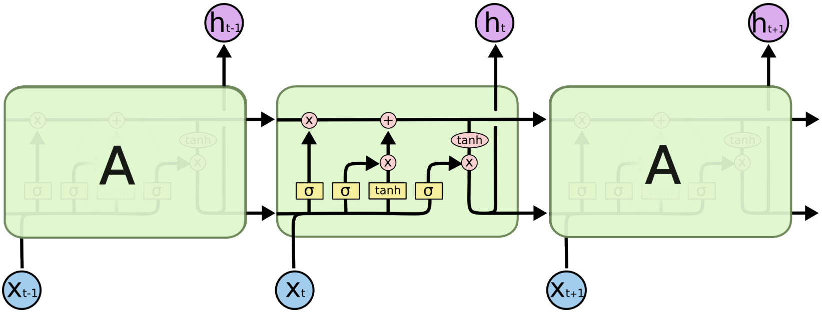
# Recurrent Neural Networks




Compute $\nabla_\Theta \log \mathbb{P}_\Theta(\mathbf{x})$ by "truncated backpropagation through time"

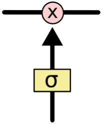
- i.e., reverse chain-rule + "clip" exploding derivatives

# Long short term memory[1]

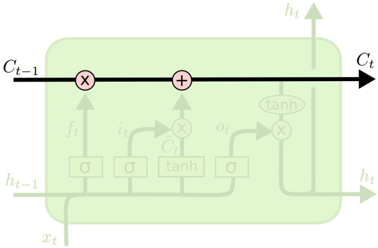



Information passes through a series of "gates"

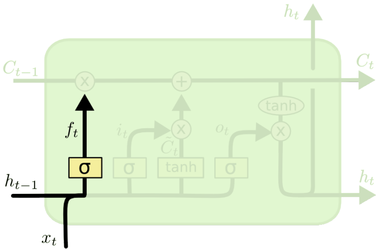
- "Gate" = multiplication with sigmoid
  - $\sigma = 0 \Rightarrow$ "let nothing thru"
  - $\sigma = 1 \Rightarrow$ "let all thru"




[1] graphics have slight differences with Graves 2013; they are taken from: https://colah.github.io/posts/2015-08-Understanding-LSTMs

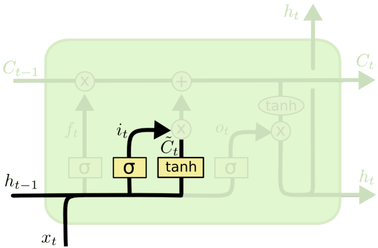



$C_t$ = "cell state" = flows horizontally across LSTM units



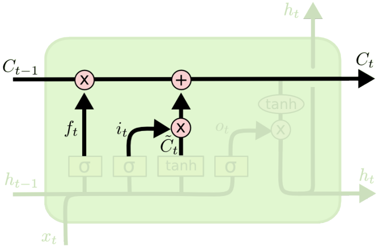

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

$f_t$ = "forget gate" = gate to forget information from $C_{t-1}$



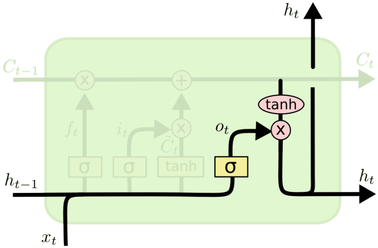

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

$i_t$ = "input gate" = gate to add information from $h_{t-1}$ and $x_t$ to $C_t$




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

Update $C_t$ using $f_t$ and $i_t$

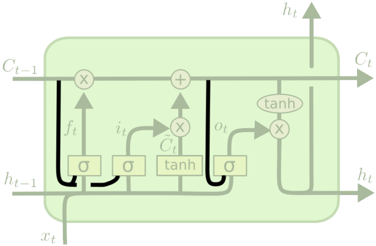



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

$o_t$ = "output gate" = gate to output information to cell above/right

Many variations on LSTM exist. Here is the one used in Graves 2013:



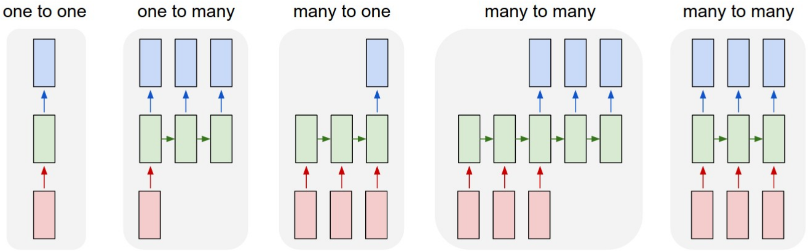

$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

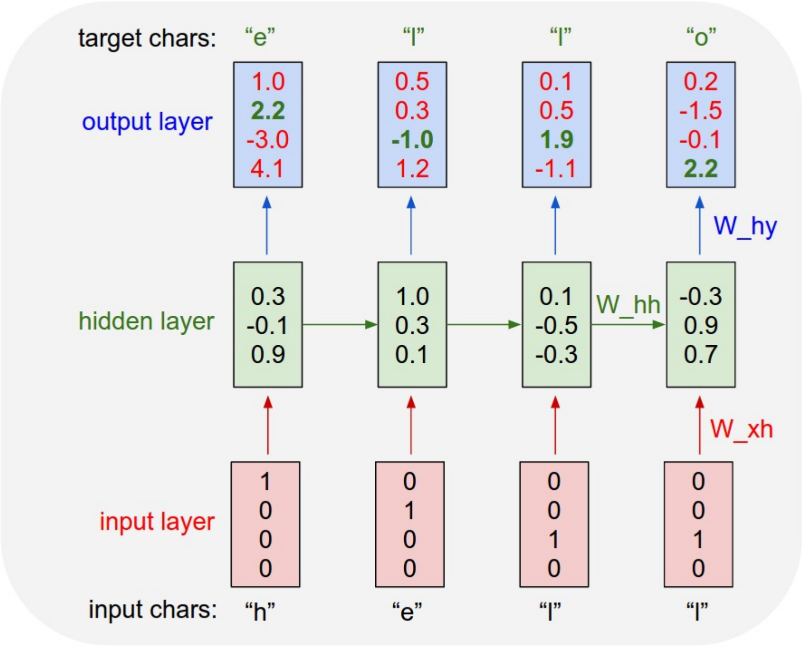
Notice the extra "peephole" connections from $C$ to $f, i, o$

# Table of Contents

# Text Prediction

How do we model a NN that takes a variable input length?
(http://karpathy.github.io/2015/05/21/rnn-effectiveness/).




target chars:
"e"
"l"
"l"
"o"
output layer
1.0 2.2 -3.0 4.1
0.5 0.3 -1.0 1.2
0.1 0.5 1.9 -1.1
0.2 -1.5 -0.1 2.2
W_hy
hidden layer
0.3 -0.1 0.9
1.0 0.3 0.1
0.1 -0.5 -0.3
-0.3 0.9 0.7
W_hh
W_xh
input layer
1 0 0 0
0 1 0 0
0 0 1 0
0 0 1 0
input chars:
"h"
"e"
"l"
"l"

# RNN and LSTM Shakespeare

We compare an RNN with an LSTM to predict Shakespeare.

# LSTM Shakespeare

```
KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

# LSTM Wikipedia

```
 Naturalism and decision for the majority of Arab countries'
capitalide was grounded  by the Irish language by [[John
Clair]], [[An Imperial Japanese Revolt]], associated  with
Guangzham's sovereignty. His generals were the powerful
ruler of the Portugal  in the [[Protestant Immineners]],
which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired
prison, training. The  emperor travelled back to [[Antioch,
Perth, October 25|21]] to note, the Kingdom  of Costa Rica,
unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]],
known  in western [[Scotland]], near Italy to the conquest
of India with the conflict.
```

# LSTM Wiki Markdown

```
{ { cite journal | id=Cerling Nonforest Department|format=
    Newlymeslated|none } }
''www.e-complete''.

'''See also''': [[List of ethical consent processing]]

== See also ==
*[[Iender dome of the ED]]
*[[Anti-autism]]

===[[Religion|Religion]]===
*[[French Writings]]
*[[Maria]]
*[[Revelation]]
*[[Mount Agamul]]
```

# LSTM XML


```
<page>
  <title>Antichrist</title>
  <id>865</id>
  <revision>
    <id>15900676</id>
    <timestamp>2002-08-03T18:14:12Z</timestamp>
    <contributor>
      <username>Paris</username>
      <id>23</id>
    </contributor>
    <minor />
    <comment>Automated conversion</comment>
    <text xml:space="preserve">#REDIRECT [[Christianity]]</
        text>
  </revision>
</page>
```

# LSTM XML


```xml
<page>
  <title>Antichrist</title>
  <id>865</id>
  <revision>
    <id>15900676</id>
    <timestamp>2002-08-03T18:14:12Z</timestamp>
    <contributor>
      <username>Paris</username>
      <id>23</id>
    </contributor>
    <minor />
    <comment>Automated conversion</comment>
    <text xml:space="preserve">#REDIRECT [[Christianity]]</text>
  </revision>
</page>
```

# LSTM LaTeX

For $\bigoplus_{n=1,\ldots,m}$ where $\mathcal{L}_{m_\bullet} = 0$, hence we can find a closed subset $\mathcal{H}$ in $\mathcal{H}$ and any sets $\mathcal{F}$ on $X$, $U$ is a closed immersion of $S$, then $U \to T$ is a separated algebraic space.

*Proof.* Proof of (1). It also start we get

$$S = \mathrm{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \to V$. Consider the maps $M$ along the set of points $Sch_{fppf}$ and $U \to U$ is the fibre category of $S$ in $U$ in Section, **??** and the fact that any $U$ affine, see Morphisms, Lemma **??**. Hence we obtain a scheme $S$ and any open subset $W \subset U$ in $Sh(G)$ such that $\mathrm{Spec}(R') \to S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that $f_i$ is of finite presentation over $S$. We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \to \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma **??** we can define a map of complexes $\mathrm{GL}_{S'}(x'/S'')$ and we win. □

To prove study we see that $\mathcal{F}|_U$ is a covering of $\mathcal{X}'$, and $\mathcal{T}_i$ is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and $\mathcal{F}_p$ exists and let $\mathcal{F}_i$ be a presheaf of $\mathcal{O}_X$-modules on $\mathcal{C}$ as a $\mathcal{F}$-module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\mathrm{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1}\mathcal{F})$$

is a unique morphism of algebraic stacks. Note that

$$\mathrm{Arrows} = (Sch/S)^{opp}_{fppf}, (Sch/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longmapsto (U, \mathrm{Spec}(A))$$

is an open subset of $X$. Thus $U$ is affine. This is a continuous map of $X$ is the inverse, the groupoid scheme $S$.

*Proof.* See discussion of sheaves of sets. □

The result for prove any open covering follows from the less of Example **??**. It may replace $S$ by $X_{spaces,\acute{e}tale}$ which gives an open subspace of $X$ and $T$ equal to $S_{Zar}$, see Descent, Lemma **??**. Namely, by Lemma **??** we see that $R$ is geometrically regular over $S$.

**Lemma 0.1.** *Assume (3) and (3) by the construction in the description.*

*Suppose $X = \lim |X|$ (by the formal open covering $X$ and a single map $\underline{Proj}_X(\mathcal{A}) = \mathrm{Spec}(B)$ over $U$ compatible with the complex*

$$Set(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

*When in this case of to show that $\mathcal{Q} \to \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition **??** (without element is when the closed subschemes are catenary. If $T$ is surjective we may assume that $T$ is connected with residue fields of $S$. Moreover there exists a closed subspace $Z \subset X$ of $X$ where $U$ in $X'$ is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem*

(1) *$f$ is locally of finite type. Since $S = \mathrm{Spec}(R)$ and $Y = \mathrm{Spec}(R)$.*

*Proof.* This is form all sheaves of sheaves on $X$. But given a scheme $U$ and a surjective étale morphism $U \to X$. Let $U \cap U = \coprod_{i=1,\ldots,n} U_i$ be the scheme $X$ over $S$ at the schemes $X_i \to X$ and $U = \lim_i X_i$. □

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{X,\ldots,0}$.

**Lemma 0.2.** *Let $X$ be a locally Noetherian scheme over $S$, $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}^n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq \mathfrak{p}$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.*

**Lemma 0.3.** *In Situation **??**. Hence we may assume $\mathfrak{q}' = 0$.*

*Proof.* We will use the property we see that $\mathfrak{p}$ is the mext functor (**??**). On the other hand, by Lemma **??** we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where $K$ is an $F$-algebra where $\delta_{n+1}$ is a scheme over $S$. □

# LSTM C++

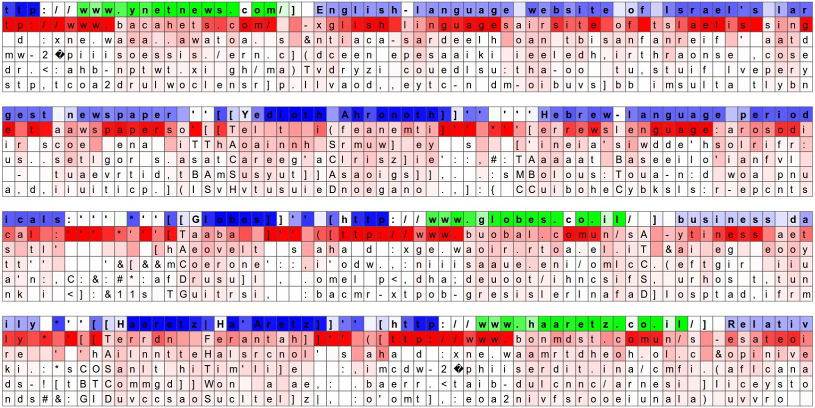
```
/*
 * If this error is set, we will need anything right after
 * that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
  unsigned long flags;
  int lel_idx_bit = e->edd, *sys & ~((unsigned long) *
      FIRST_COMPAT);
  buf[0] = 0xFFFFFFFF & (bit << 4);
  min(inc, slist->bytes);
  printk(KERN_WARNING "Memory allocated %02x/%02x, "
    "original MLL instead\n"),
    min(min(multi_run - s->len, max) * num_data_in),
    frame_pos, sz + first_seg);
  div_u64_w(val, inb_p);
  spin_unlock(&disk->queue_lock);
  mutex_unlock(&s->sock->mutex);
  mutex_unlock(&func->mutex);
  return disassemble(info->pending_bh);
}
```

# LSTM C++

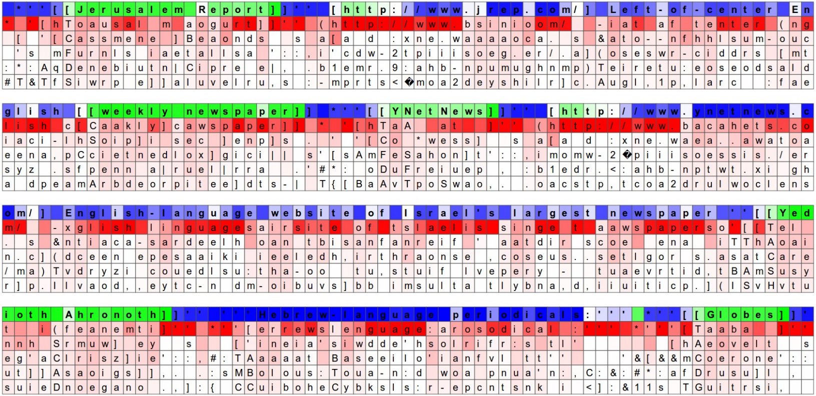
```
static void num_serial_settings(struct tty_struct *tty)
{
  if (tty == tty)
    disable_single_st_p(dev);
  pci_disable_spool(port);
  return 0;
}

static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

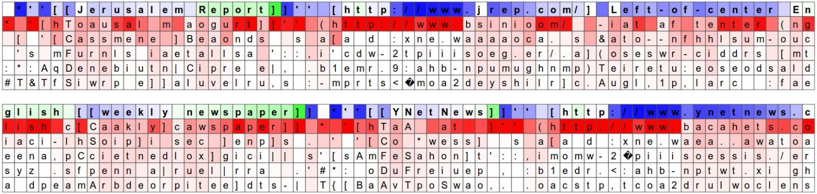
# Visualizing Predictions: What Do Neurons Do?



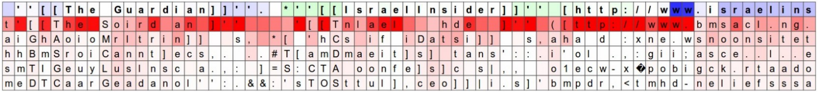
# Visualizing Predictions: What Do Neurons Do?



# Visualizing Predictions: What Do Neurons Do?



# Visualizing Predictions: What Do Neurons Do?



# Visualizing Predictions: What Do Neurons Do?

Cell sensitive to position in line:

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.

Cell that turns on inside quotes:

"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."

# Visualizing Predictions: What Do Neurons Do?

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
   siginfo_t *info)
{
 int sig = next_signal(pending, mask);
 if (sig) {
  if (current->notifier) {
   if (sigismember(current->notifier_mask, sig)) {
    if (!(current->notifier)(current->notifier_data)) {
     clear_thread_flag(TIF_SIGPENDING);
     return 0;
    }
   }
  }
  collect_signal(sig, pending, info);
 }
 return sig;
}
```

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
```

# Visualizing Predictions: What Do Neurons Do?

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information.  The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
          struct audit_field *sf)
{
 int ret = 0;
 char *lsm_str;
 /* our own copy of lsm_str */
 lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
 if (unlikely(!lsm_str))
  return -ENOMEM;
 df->lsm_str = lsm_str;
 /* our own (refreshed) copy of lsm_rule */
 ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
          (void **)&df->lsm_rule);
 /* Keep currently invalid fields around in case they
  * become valid after a policy reload. */
 if (ret == -EINVAL) {
  pr_warn("audit rule for LSM \'%s\' is invalid\n",
   df->lsm_str);
  ret = 0;
 }
 return ret;
}
```

# Visualizing Predictions: What Do Neurons Do?

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some ")":

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
 char *str;
 if (!*bufp || (len == 0) || (len > *remain))
  return ERR_PTR(-EINVAL);
 /* Of the currently implemented string fields, PATH_MAX
  * defines the longest valid length.
  */
 if (len > PATH_MAX)
  return ERR_PTR(-ENAMETOOLONG);
 str = kmalloc(len + 1, GFP_KERNEL);
 if (unlikely(!str))
  return ERR_PTR(-ENOMEM);
 memcpy(str, *bufp, len);
 str[len] = 0;
 *bufp += len;
 *remain -= len;
 return str;
}
```

# Table of Contents

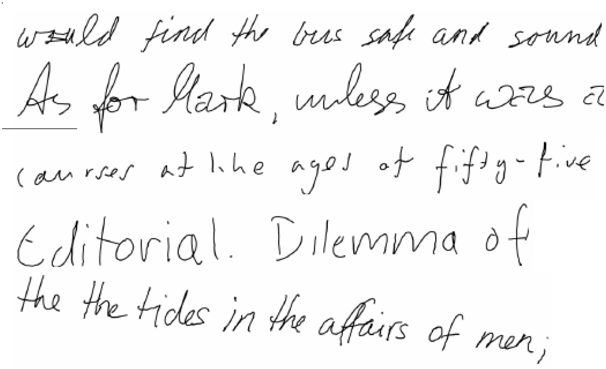



Figure: Training samples from IAM handwriting database

Input $x_t = (\mathrm{pos}_t - \mathrm{pos}_{t-1}) \times (\text{is-end-of-stroke}) \in \mathbb{R}^2 \times \{0, 1\}$

# Mixture density network

$$x_t \in \mathbb{R} \times \mathbb{R} \times \{0, 1\}$$

$$y_t = (e_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^M)$$

$$\mathbb{P}(x_{t+1} \mid y_t) = \sum_{j=1}^{M} \pi_t^j \mathcal{N}(x_{t+1} \mid \mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} e_t & \text{if } (x_{t+1})_3 = 1 \\ 1 - e_t & \text{else} \end{cases}$$

- $e_t$ = end-of-stroke prob = $\frac{1}{1+\exp(\hat{e}_t)}$
- $\pi_t^j$ = mixture prob = $\frac{\exp(\hat{\pi}_t^j)}{\sum_{j'} \exp(\hat{\pi}_t^{j'})}$
- $\mu_t^j$ = component mean = $\hat{\mu}_t^j$
- $\sigma_t^j$ = component variance = $\exp(\hat{\sigma}_t^j)$
- $\rho_t^j$ = component x/y correlation = $\tanh(\hat{\rho}_t^j)$

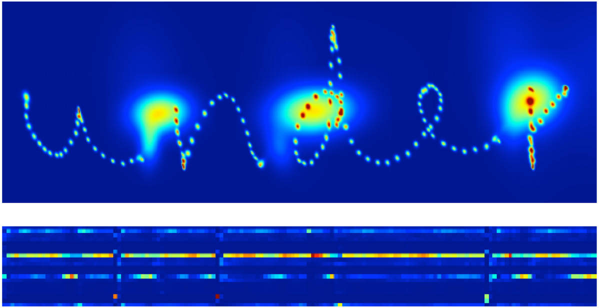


Figure: Mixture density outputs as word “under” is written

- Blobs = Predictions at end of strokes for first point in next stroke
- Lower panel = component weights

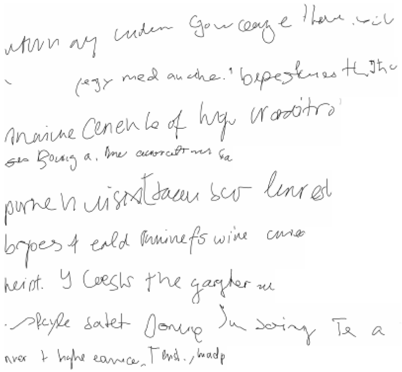
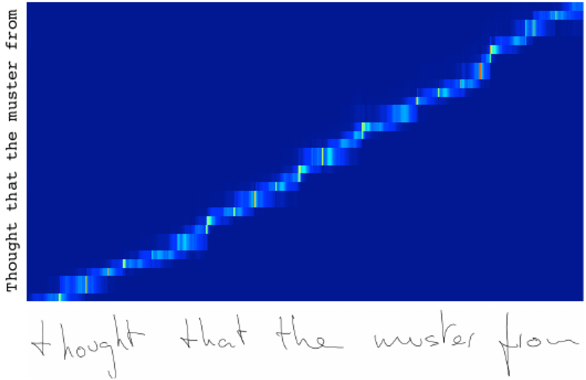


Figure: Samples generated by prediction network (700-timesteps each). Network has learned strokes, some characters, and even short words (e.g. "of").

- 3 hidden layers, each with 400 LSTM cells
- $3.4 \times 10^6$ parameters total
- Trained with rmsprop and adaptive weight noise
- Output derivatives $\frac{\partial \log \mathbb{P}(\mathbf{x})}{\partial \hat{y}_t}$ clipped to be in $[-100, 100]$; LSTM derivatives clipped within $[-10, 10]$
  - Clipping vital for numerical stability

# Handwriting synthesis




- Generate handwriting for a given text
- Need to align handwriting with discrete text
- Bottom layer → distribution of current position ("soft window")

# Synthesis Network

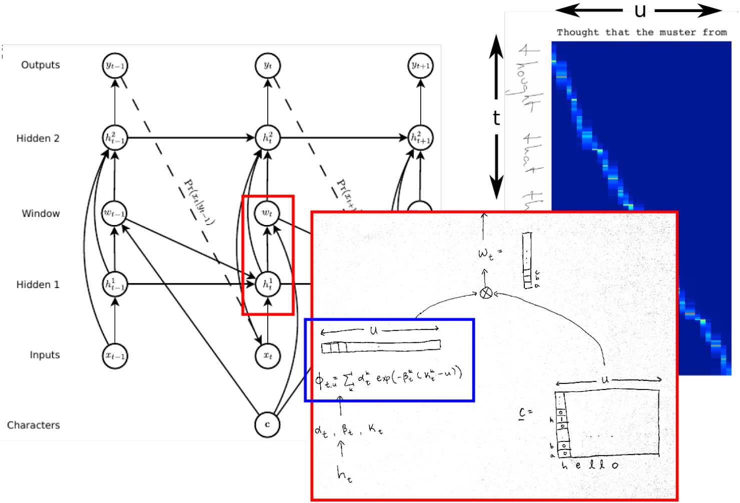
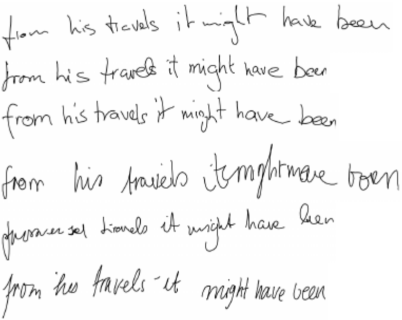
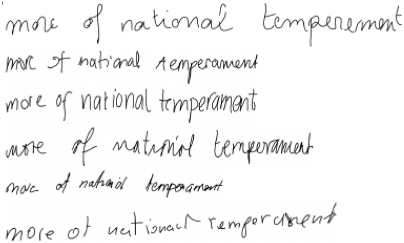



Figure: $\phi_t(h_t^1)$ = distn over positions; $w_t = \mathbf{c}\phi_t$ = distn over characters

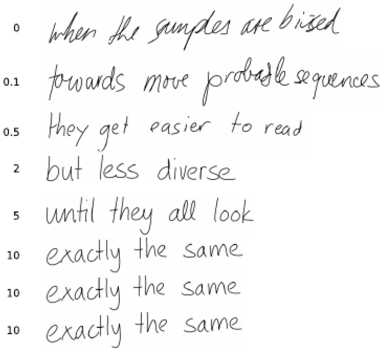
# Unbiased sampling





- Sample from $\mathbb{P}(\mathbf{x} \mid \mathbf{c})$ by iteratively sampling from $\mathbb{P}(x_{t+1} \mid y_t)$
- Stop when $\phi(t, U+1) > \max_{u \leq U} \phi(t, u)$
- First line of each block is real; subsequent lines generated by network

# Biased sampling



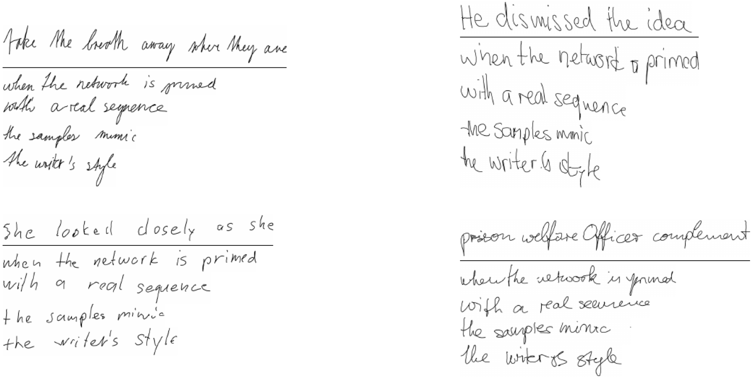

- $\downarrow$ variance $\Rightarrow$ neater writing
  - Component variance:

$$\sigma_t^j = \exp(\hat{\sigma}_t^j - b)$$

  - Mixture probabilities:

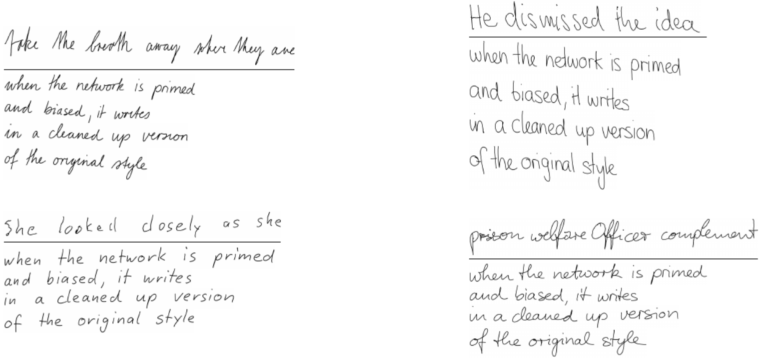
$$\pi_t^j = \frac{\exp(\hat{\pi}_t^j(1+b))}{\sum_{j'} \exp(\hat{\pi}_t^{j'}(1+b))}$$

# Primed sampling




- Mimic specific person without retraining whole network
- For real $\mathbf{x}, \mathbf{c}$ and new $\mathbf{s}$, set $\mathbf{c}' = \mathbf{c} + \mathbf{s}$, and clamp first $T$ inputs to $\mathbf{x}$

# Combining biasing + priming




Application: artificial enhancement of poor handwriting?