



Generating Sequences With Recurrent Neural Networks

Dealers choice:

Sanchit Arora (2019101047)

Jatin Gupta (2021201048)

Sashank S (2021701038)

01.

**Introduction &
Proposal**

02.

Project Expectations

03.

**Implementation
Details**

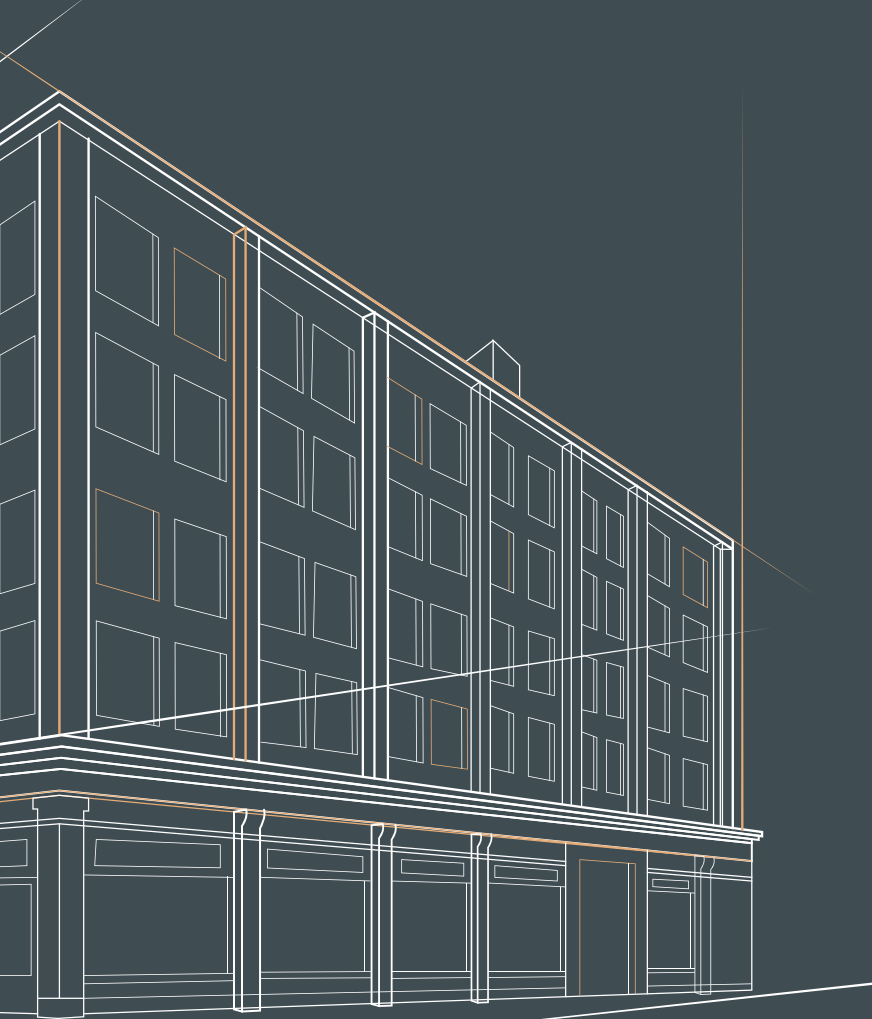
04.

Results

05.

Future Works





01

Introduction & Proposal



• • • • •

comes
next.

Long Short-term Memory (LSTM)
better at storing and accessing
information than standard RNNs.

A 3x10 grid of dots, consisting of 3 rows and 10 columns of small black dots.

A 4x3 grid of dots, consisting of 12 dots arranged in 4 rows and 3 columns.

Goals

Text Prediction

Task: Generate text sequences one character at a time.

Either at **Character** level or **Word** level.

Datasets Used : **Penn Treebank** and **Wikipedia**

Handwriting Prediction

Task: Generate pen trajectories by predicting one (x,y) point at a time.

Data: **IAM online handwriting**

Handwriting Synthesis

Task: Want to tell the network what to write without losing the distribution over how it writes.

Can do this by conditioning the predictions on a text sequence.



Generating Sequences




How to generate sequences?

- Obvious way is to try and predict what happens next

$$\Pr(\mathbf{x}) = \prod_t \Pr(x_t | x_{1:t-1})$$

Even for humans, it is easier to forecast how *likely* certain events will be in the future compared to predicting the future exactly.






Use of Memory



Need to remember the past to predict the future.

Advt of Long Memory:

- Can store and generate longer range patterns
 - Robust to Mistakes
 - Analyse “Disconnected” patterns
- 

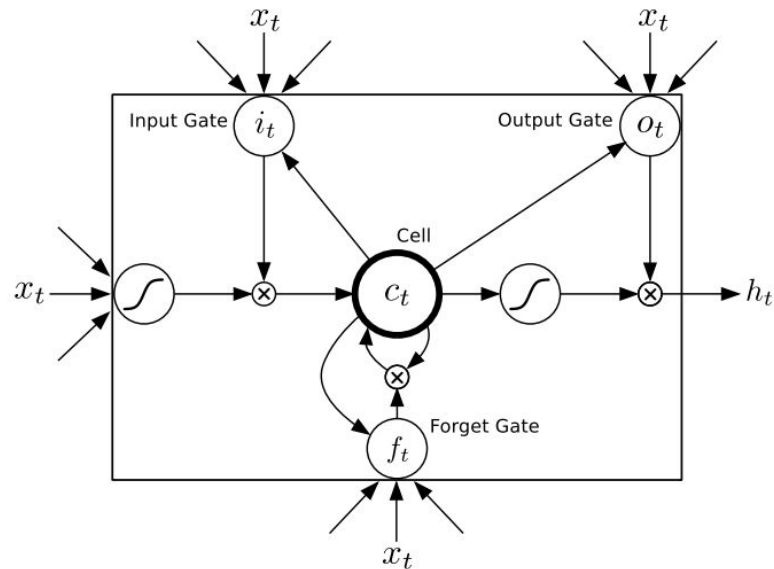
Long Short Term Memory

- **LSTM** is an RNN architecture designed to have a better memory. It uses linear memory cells surrounded by multiplicative gate units to store read, write and reset information.

Input gate: scales input to cell (write)

Output gate: scales output from cell (read)

Forget gate: scales old cell value (reset)

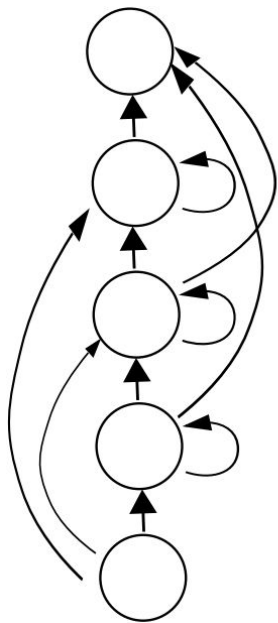


Architecture

output
layer

LSTM
layers

inputs



- Deep Recurrent LSTM network
- Inputs arrive one at a time, outputs determine predictive distribution over next input
- Train by minimising log-loss

$$\sum_{t=1}^T -\log \Pr(x_t | x_{1:t-1})$$

- Generate by sampling from output distribution and feeding into input

Text Prediction

output distribution

$$\hat{y}^{(t)} = \text{softmax}(U h^{(t)} + b_2) \in \mathbb{R}^{|V|}$$

hidden states

$$h^{(t)} = \sigma(W_h h^{(t-1)} + W_e e^{(t)} + b_1)$$

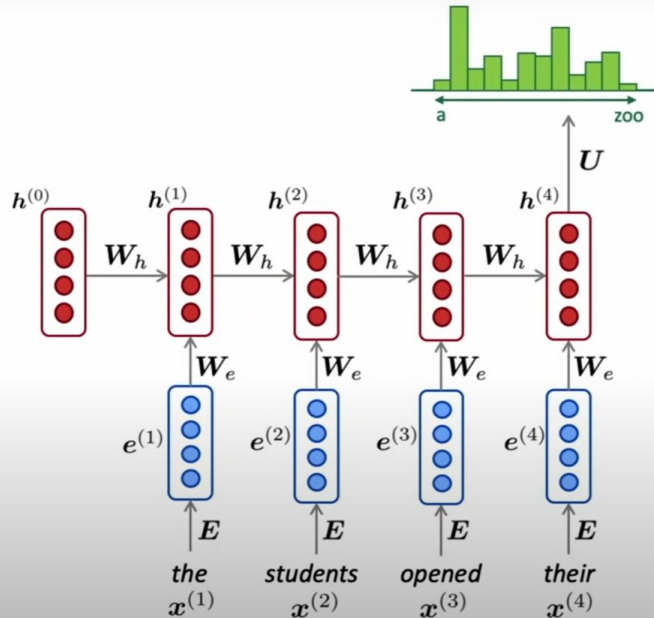
$h^{(0)}$ is the initial hidden state

word embeddings

$$e^{(t)} = E x^{(t)}$$

words / one-hot vectors

$$x^{(t)} \in \mathbb{R}^{|V|}$$



Language modeling is the task of predicting what word comes next.

Given a sequence of words $x(1), x(2), \dots, x(t)$, compute the probability distribution of the next word $x(t+1)$.

Disadvts of Word based Models:

- The number of words often exceeds 100,000.
- Many parameters to model.
- Huge amount of training data.
- High computational cost of evaluating softmax models.

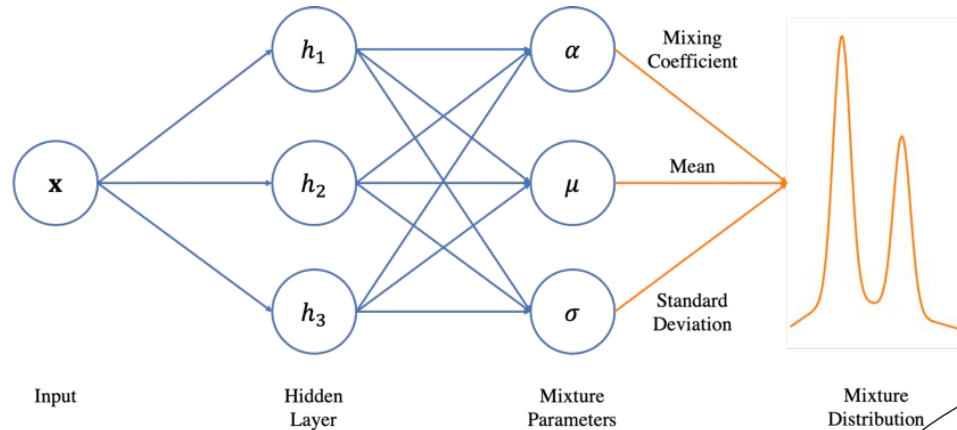
Adopt Character based Models.

Handwriting Prediction

The principal challenge in applying a prediction network to handwriting data was determining a predictive distribution suitable for real-valued inputs.

Mixture Density Networks:

Combine a deep neural network (DNN) and a mixture of distributions. The DNN provides the parameters for multiple distributions, which are then mixed by some weights. These weights are also provided by the DNN. The resulting (multimodal) conditional probability distribution helps us to model complex patterns found in real-world data.

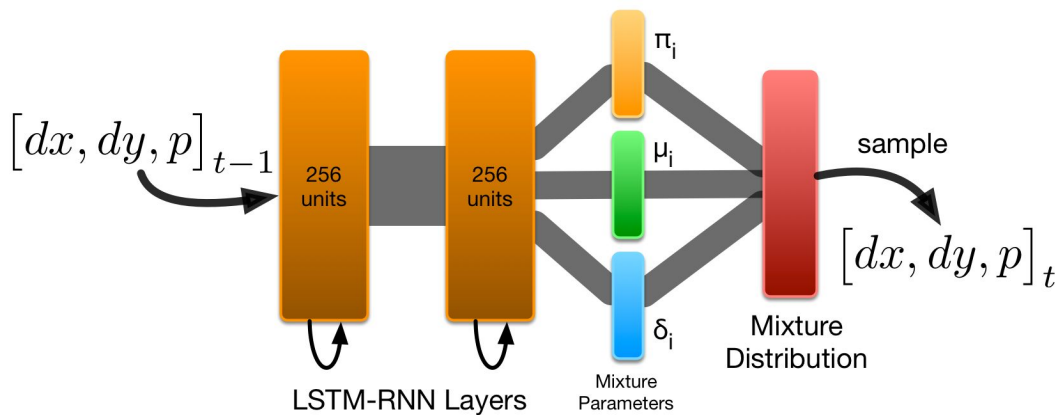


MDNs with RNNs

The output distribution is conditioned not only on the current input, but on the history of previous inputs.

Why do we need a 3D mixture model?

One dimension for pen-X, one for pen-Y, and one for pen-UpDown.

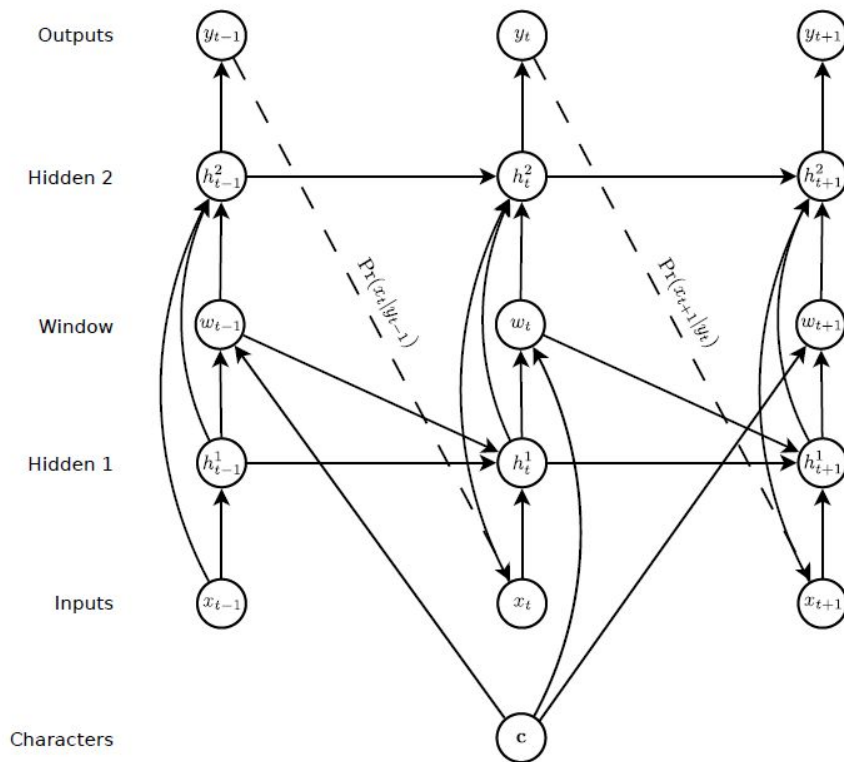


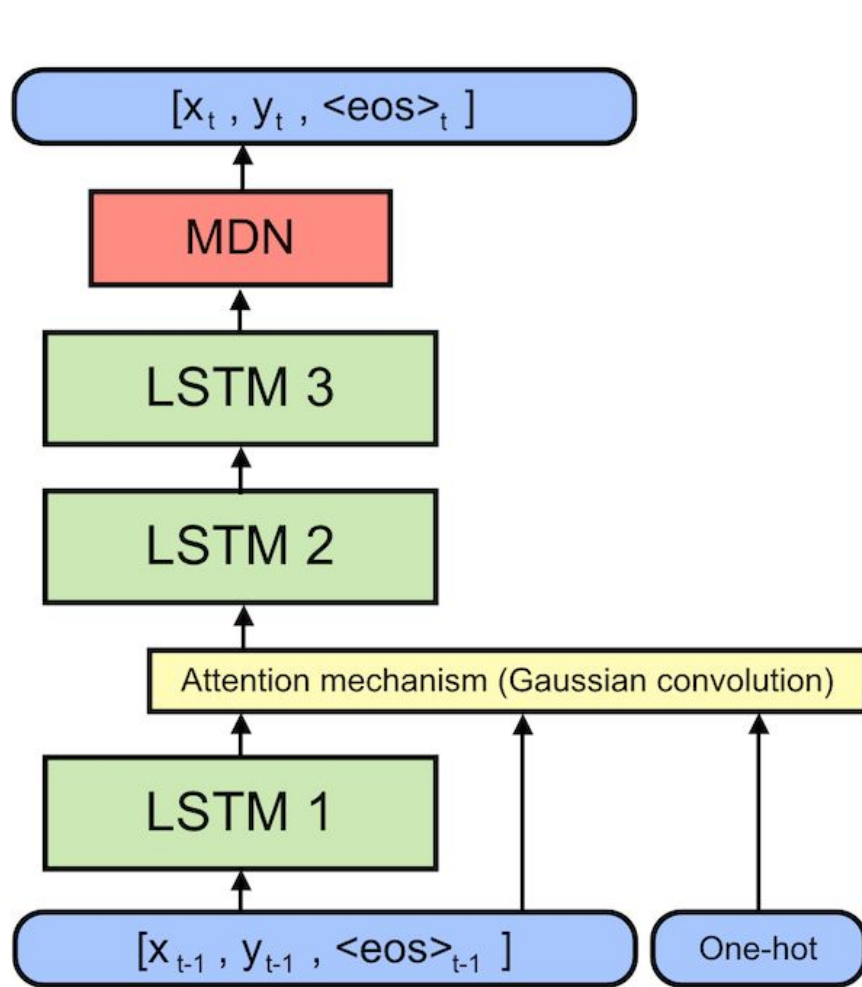
Handwriting Synthesis

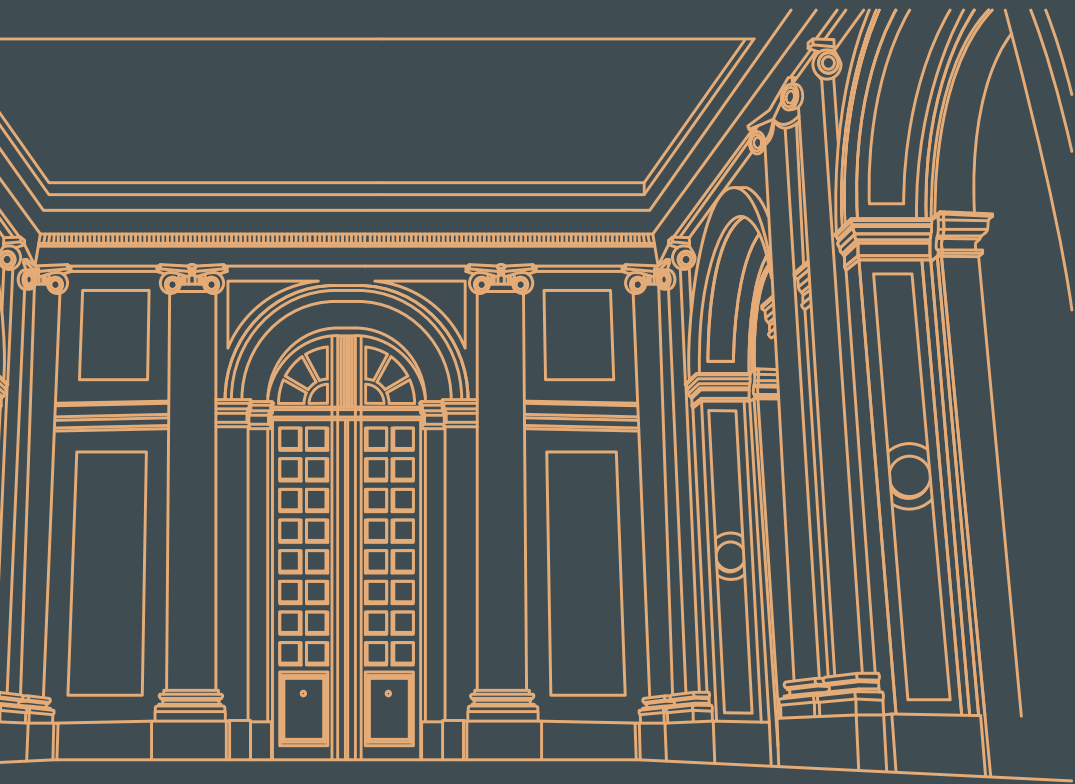
Handwriting synthesis is the generation of handwriting for a given text.

Prediction network to generate data sequences conditioned on some high-level annotation sequence.

A *soft window* is convolved with the text string and fed in as an extra input to the prediction network.







02

Project Expectations





Expected Deliverables




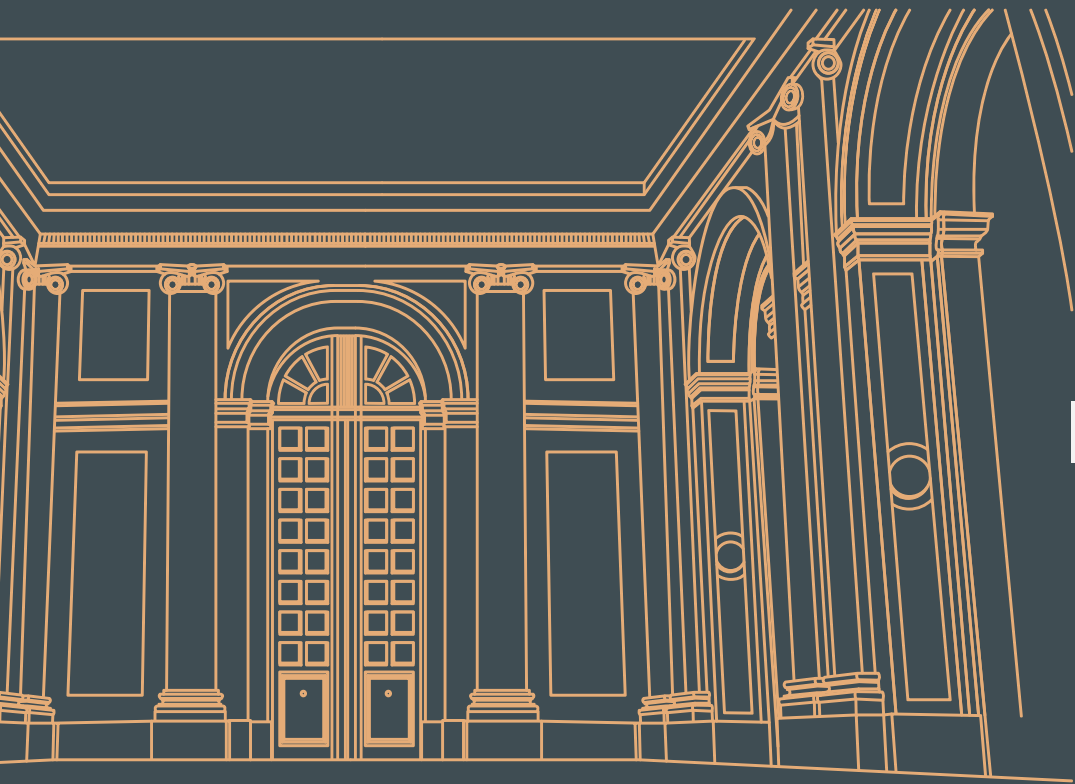
Build a prediction network for Penn Treebank dataset and perform character and word level experiments.

Build a prediction network for Hutter prize Wiki dataset and generate a sample wikipedia document.

Prediction network can be applied to real-valued data through the use of a mixture density output layer. Generate handwriting samples from the Handwriting Database.

Extend the prediction network that allows it to condition its outputs on a short annotation sequence whose alignment with the predictions is unknown, making it suitable for handwriting synthesis.





03

Implementation ● Details

Text Prediction



Penn Treebank Dataset



Character Level

- Number of tokens

Train: 5017483

Valid: 393043

Test: 442424

Word Level


- Number of tokens

Train: 929589

Valid: 73760

Test: 82430

The vocabulary is limited to 10,000 words, with all other words mapped to a special ‘unknown word’ token. The end-of-sentence token was included in the input sequences, and was counted in the sequence loss.





Encoder-Decoder Network


Character Level

```
RNNModel(  
  (drop): Dropout(p=0.5, inplace=False)  
  (encoder): Embedding(50, 256)  
  (rnn): LSTM(256, 1000, dropout=0.5)  
  (decoder): Linear(in_features=1000, out_features=50, bias=True)  
)
```

- Single hidden layer with 1000 LSTM units.
- The input and output layers were size 50, giving approximately 4.3M weights in total
- All networks were trained with stochastic gradient descent, using a learn rate of 0.0001 and a momentum of 0.99.
- The LSTM derivatives were clipped in the range $[-1, 1]$

Word Level

```
RNNModel(  
  (drop): Dropout(p=0.5, inplace=False)  
  (encoder): Embedding(10000, 256)  
  (rnn): LSTM(256, 1000, dropout=0.5)  
  (decoder): Linear(in_features=1000, out_features=10000, bias=True)  
)
```

- Single hidden layer with 1000 LSTM units.
 - 10,000 inputs and outputs and around 54M weights.
- 




Regularization



Method 1: Weight noise with a std. deviation of 0.075 applied to the network weights at the start of each training sequence.

Method 2: Adaptive weight noise, where the variance of the noise is learned along with the weights using a Variational inference loss function.

When weight noise was used, the network was initialised with the final weights of the unregularised network. Similarly, when adaptive weight noise was used, the weights were initialised with those of the network trained with weight noise.





Wikipedia Dataset



Data: Raw wikipedia markup from Hutter challenge (100 MB)

Total number of articles: 5824596

Unique characters: 621 (Vocab Size)

Sequence Length = 200



Network Used

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(64, None, 256)	158976
lstm (LSTM)	(64, None, 1024)	5246976
dense (Dense)	(64, None, 621)	636525
Total params: 6,042,477		
Trainable params: 6,042,477		
Non-trainable params: 0		

- Single hidden layer with 1024 LSTM units.
- Input Embedding Layer with Vocab Size 621.
- Output Dense Layer with Vocab Size 621.
- Network trained with Adam Optimizer, using a learn rate of 0.001.

Handwriting Prediction




IAM-OnDB



Data: IAM online handwriting database (IAM-OnDB)

10K training sequences, many writers, unconstrained style, captured from whiteboard.

IAM-OnDB is divided into a training set, two validation sets and a test set, containing respectively 5364, 1438, 1518 and 3859 handwritten lines taken from 775, 192, 216 and 544 forms.






Network Used



```
HandwritingGenerationModel(  
  (lstm): LSTM(3, 256, num_layers=3, dropout=0.2)  
  (z_e): Linear(in_features=256, out_features=1, bias=True)  
  (z_pi): Linear(in_features=256, out_features=10, bias=True)  
  (z_mu1): Linear(in_features=256, out_features=10, bias=True)  
  (z_mu2): Linear(in_features=256, out_features=10, bias=True)  
  (z_sigma1): Linear(in_features=256, out_features=10, bias=True)  
  (z_sigma2): Linear(in_features=256, out_features=10, bias=True)  
  (z_rho): Linear(in_features=256, out_features=10, bias=True)  
)
```


- 3 inputs: Δx , Δy , pen up/down
 - 61 output units
 - 10 two dimensional Gaussians for $x, y = 20$ means (linear : z_mu1, z_mu2) + 20 std. devs (exp: z_sigma1, z_sigma2) + 10 correlations (tanh : z_rho) + 10 weights (softmax : z_pi) • 1 sigmoid (z_e) for up/down
 - 1 hidden Layer, 256 LSTM cells
 - Adam : $lr=0.005$
 - Error clipped during backward pass
- 

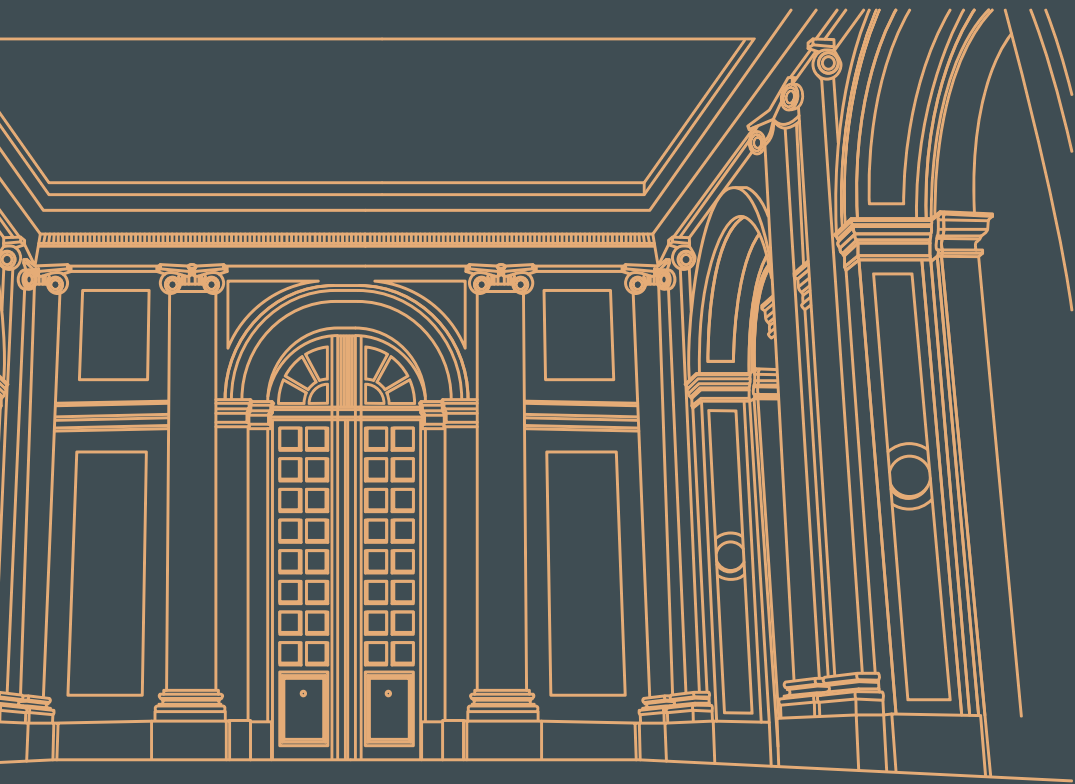
Handwriting Synthesis



Network Used

```
LSTMSynthesis(  
  (lstm1): LSTM1(  
    (lstm): LSTMCell(63, 400)  
    (window): Window(  
      (linear): Linear(in_features=400, out_features=60, bias=True)  
    )  
  )  
  (lstm2): LSTM2(  
    (lstm): LSTM(463, 400, batch_first=True)  
  )  
  (linear): Linear(in_features=800, out_features=121, bias=True)  
  (tanh): Tanh()  
)
```

- The character sequence is encoded with one-hot vectors in the **Window** layer : Vocab Size = 60
 - 2 Hidden LSTM Layers with size 400
 - MDN with 121 output units : 20 two dimensional Gaussians for x,y = 40 means (linear) + 40 std. devs (exp) + 20 correlations (tanh) + 20 weights (softmax) • 1 sigmoid for up/down
- 



04

Results





Evaluation for Text Prediction



Perplexity (PPL): Tells us how well our models are at predicting the next word in a text.

$$\text{PPL} = e^{L_e \cdot \bar{n}}$$

We calculate it by using the output of our loss function, the negative log likelihood L_e . \bar{n} is the average number of atomic units that the words in the test set are split into.

Bits-per-character (BPC): Describes the average character-wise cross-entropy. For the character-based model this is equal to the negative log likelihood L_2 to the base 2. For the word-based models, we need to break this value down to apply to single characters.

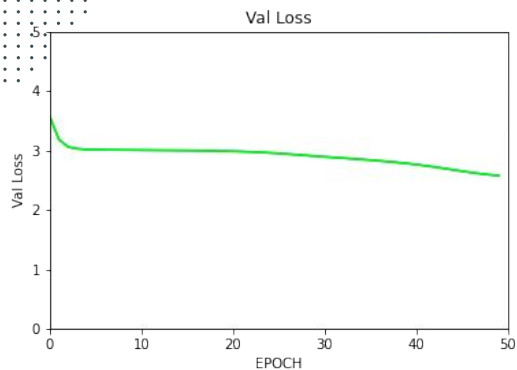
$$\text{BPC} = \frac{L_2 \cdot \bar{n}}{\bar{w} + 1}$$

Where w is the average word-length in the testset.

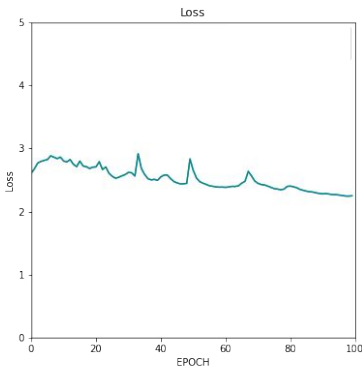


Validation Loss Curves

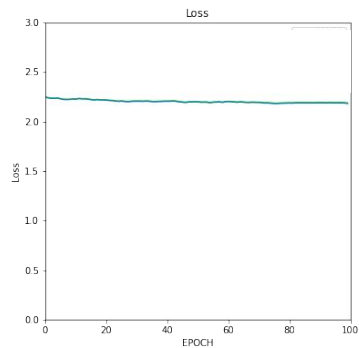
Char with No Regularization



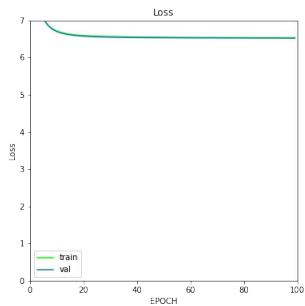
Char with Weight Noise Regularization



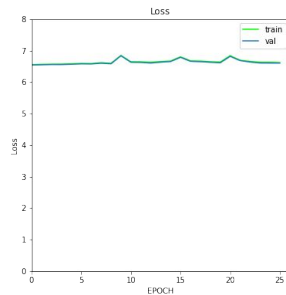
Char with Adaptive Regularization



Word with No Regularization



Word with Noise Regularization



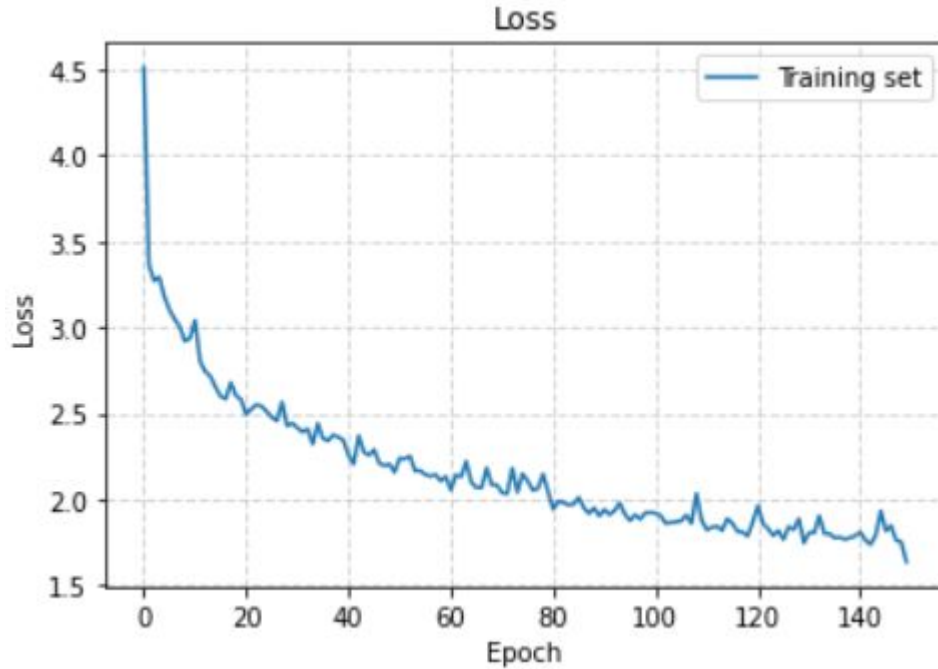


Penn Tree Bank Prediction Network



Input	Regularization	BPC	Perplexity	Epochs
Char	None	3.706	13.05	25
Char	Weight Noise	3.205	9.22	50
Char	Adaptive Weight Noise	3.113	8.65	100
Word	None	9.329	643.01	100
Word	Weight Noise	9.530	739.29	25

Wikipedia Prediction network



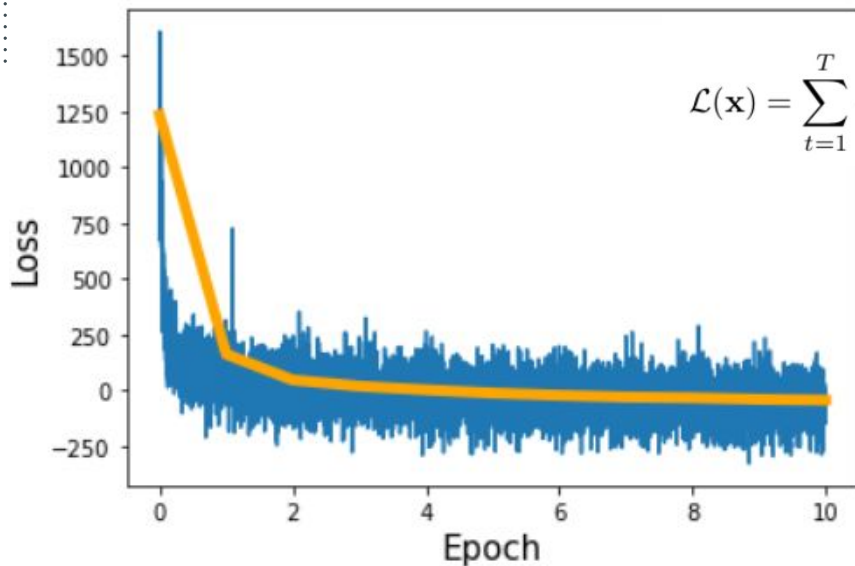
Sparse Categorical Cross Entropy

$$J(\mathbf{w}) = -\frac{1}{N} \sum_{i=1}^N [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Input: Science is

Output Prediction: Science is the official came served to the time competition in 1995, the appearly was the member of the competitors of the party of grave and of the many of the started on the cominations of the dames of the castary of the presenting in the must of the consecured to the entroding poster was the construction

Handwriting Prediction Network



In orange is the average loss per epoch, in blue the loss per batch.

$$\mathcal{L}(\mathbf{x}) = \sum_{t=1}^T -\log \left(\sum_j \pi_t^j \mathcal{N}(x_{t+1} | \mu_t^j, \sigma_t^j, \rho_t^j) \right) - \begin{cases} \log e_t & \text{if } (x_{t+1})_3 = 1 \\ \log(1 - e_t) & \text{otherwise} \end{cases}$$

Maximize the likelihood of the estimated bivariate normal distributions and Bernoulli distribution.

Output Prediction

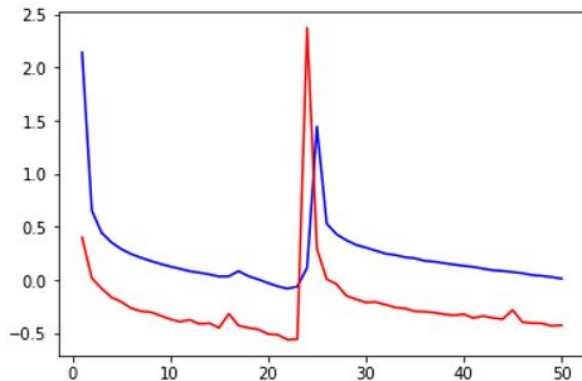
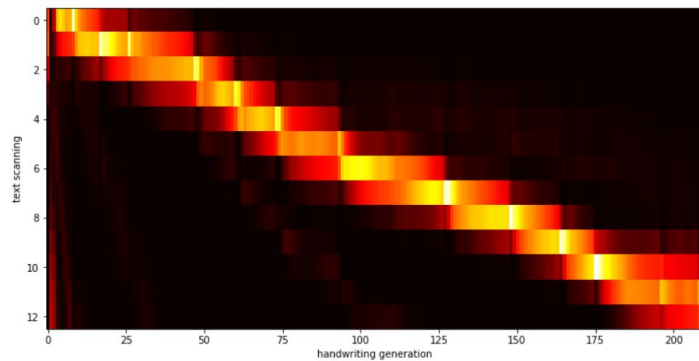
be 1/2 4 WS 1

Handwriting Synthesis Network

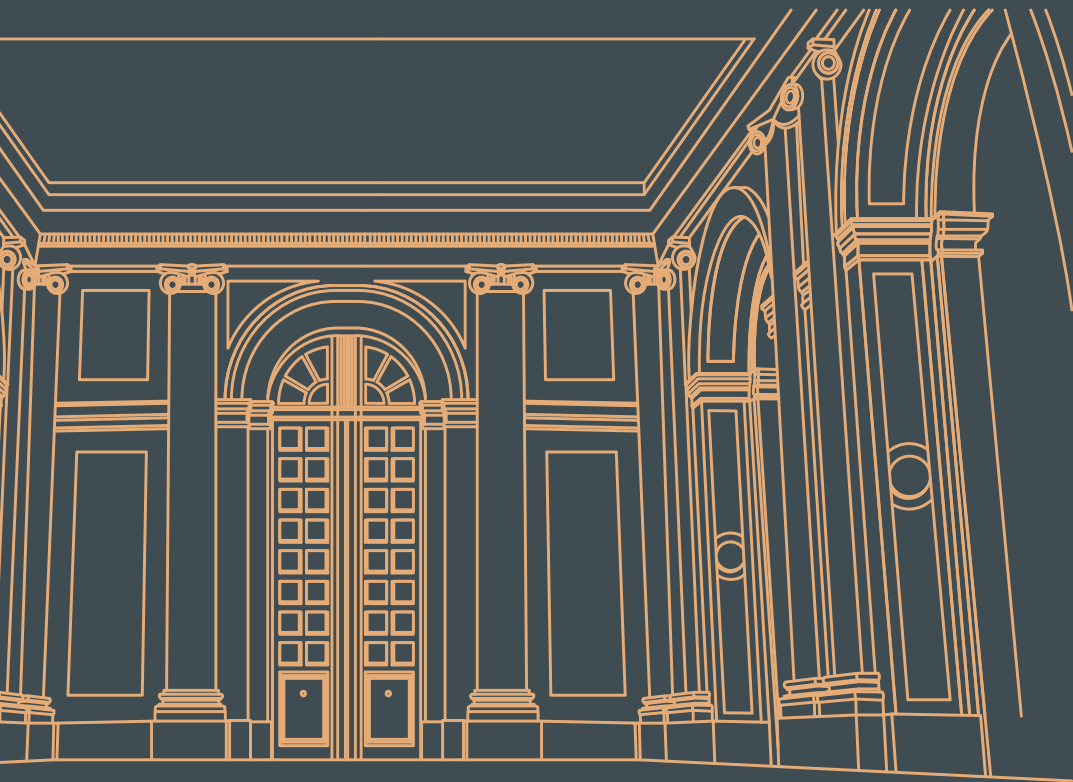
Input : Hello World

Output Synthesis

Hello World



In Blue is the training loss per epoch, in red the validation loss per epoch.



05


Future Steps





Next Steps



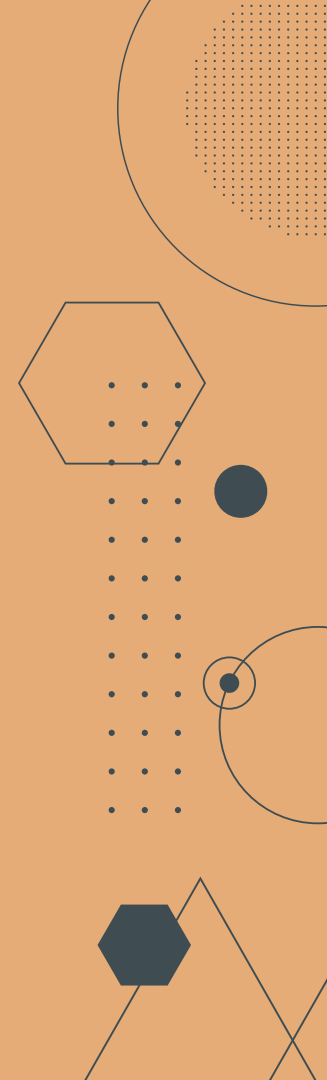
- Better understanding of internal representation. Provide more explainability.
 - Learn high level features (strokes, letters, words...) rather than adding them manually.
 - Explore other architectures for generative networks such as GANs.
- 

A vertical orange sidebar on the left side of the slide. It features a large circle with a small dark dot inside, a square with a diagonal line and a dotted pattern, a thick dark diagonal line, a small dark circle, and a series of dots arranged in a grid pattern.

Thanks!

...

CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.

A vertical orange sidebar on the right side of the slide. It features a large circle with a dotted pattern inside, a hexagon with a dotted pattern inside, a small dark circle, a series of dots arranged in a grid pattern, a small circle with a dark dot inside, and a hexagon with a dark fill.