

EFFICIENT TEXT CLASSIFICATION WITH ECHO STATE NETWORKS

Jérémie Cabessa

Playtika

Lausanne, Switzerland

&

Department of Mathematical Economics

University Paris II, France

Tübingen, 11 October, 2021

INTRODUCTION

- We consider Echo State Networks (ESNs) in the context of Natural Language Processing (NLP).
- More specifically, we introduce ESNs with pre-trained word embeddings as inputs (GloVe, BERT) for text classification.
- We show that ESNs are efficient candidates for text classification tasks.
- Some works about text classification with ESNs have already been done, but along different lines (different training paradigms).

INTRODUCTION

- We consider Echo State Networks (ESNs) in the context of Natural Language Processing (NLP).
- More specifically, we introduce ESNs with pre-trained word embeddings as inputs (GloVe, BERT) for text classification.
- We show that ESNs are efficient candidates for text classification tasks.
- Some works about text classification with ESNs have already been done, but along different lines (different training paradigms).



INTRODUCTION

- We consider Echo State Networks (ESNs) in the context of Natural Language Processing (NLP).
- More specifically, we introduce ESNs with pre-trained word embeddings as inputs (GloVe, BERT) for text classification.
- We show that ESNs are efficient candidates for text classification tasks.
- Some works about text classification with ESNs have already been done, but along different lines (different training paradigms).



INTRODUCTION

- We consider Echo State Networks (ESNs) in the context of Natural Language Processing (NLP).
- More specifically, we introduce ESNs with pre-trained word embeddings as inputs (GloVe, BERT) for text classification.
- We show that ESNs are efficient candidates for text classification tasks.
- Some works about text classification with ESNs have already been done, but along different lines (different training paradigms).

DATASETS

- **TREC-6 / TREC-50.** Question classification with 6/50 classes.
Train set / Test set: 5452 / 500 examples.
- **SST-2.** Sentiment classification with 2 classes.
Train set / Test set: 67349 / 1821 examples.
- **IMDb.** Sentiment classification with 2 classes.
Train set / Test set: 25000 / 25,000 example.
- **AG News.** Topic classification with 4 classes.
Train set / Test set: 120000 / 7600 examples.

DATASETS

- **TREC-6 / TREC-50.** Question classification with 6/50 classes.
Train set / Test set: 5452 / 500 examples.
- **SST-2.** Sentiment classification with 2 classes.
Train set / Test set: 67349 / 1821 examples.
- **IMDb.** Sentiment classification with 2 classes.
Train set / Test set: 25000 / 25,000 example.
- **AG News.** Topic classification with 4 classes.
Train set / Test set: 120000 / 7600 examples.

DATASETS

- **TREC-6 / TREC-50.** Question classification with 6/50 classes.
Train set / Test set: 5452 / 500 examples.
- **SST-2.** Sentiment classification with 2 classes.
Train set / Test set: 67349 / 1821 examples.
- **IMDb.** Sentiment classification with 2 classes.
Train set / Test set: 25000 / 25,000 example.
- **AG News.** Topic classification with 4 classes.
Train set / Test set: 120000 / 7600 examples.

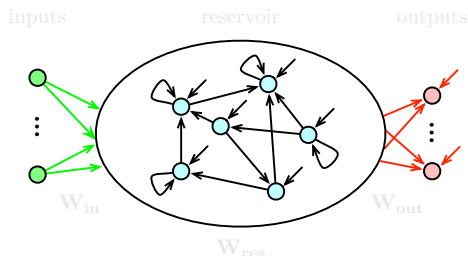
DATASETS

- **TREC-6 / TREC-50.** Question classification with 6/50 classes.
Train set / Test set: 5452 / 500 examples.
- **SST-2.** Sentiment classification with 2 classes.
Train set / Test set: 67349 / 1821 examples.
- **IMDb.** Sentiment classification with 2 classes.
Train set / Test set: 25000 / 25,000 example.
- **AG News.** Topic classification with 4 classes.
Train set / Test set: 120000 / 7600 examples.

ECHO STATE NETWORKS

Echo State Networks (ESNs) are specific kinds of recurrent neural networks. An ESN consists of:

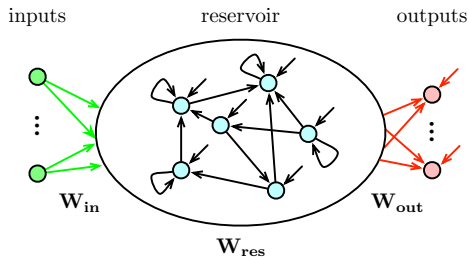
- An input layer
- A reservoir of neurons: random, recurrent and sparse
- An output layer
- Connections between layers



ECHO STATE NETWORKS

Echo State Networks (ESNs) are specific kinds of recurrent neural networks. An ESN consists of:

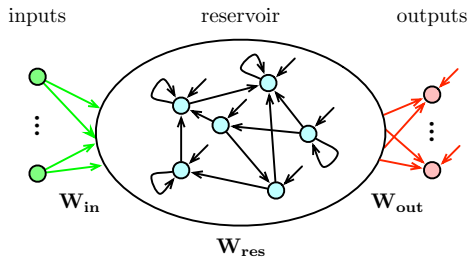
- An *input layer*
 - A *reservoir* of neurons: random, recurrent and sparse
 - An *output layer*
- ★ Only the output weights are trained!



ECHO STATE NETWORKS

Echo State Networks (ESNs) are specific kinds of recurrent neural networks. An ESN consists of:

- An *input layer*
- A *reservoir* of neurons: random, recurrent and sparse
- An *output layer*
- ★ Only the output weights are trained!

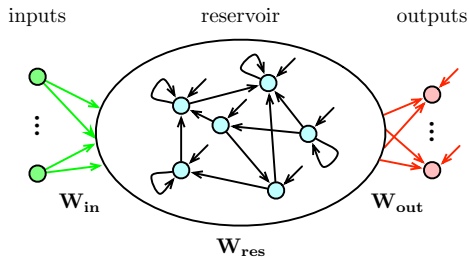


ECHO STATE NETWORKS

Echo State Networks (ESNs) are specific kinds of recurrent neural networks. An ESN consists of:

- An *input layer*
- A *reservoir* of neurons: random, recurrent and sparse
- An *output layer*

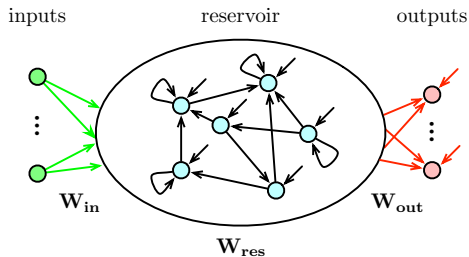
★ Only the output weights are trained!



ECHO STATE NETWORKS

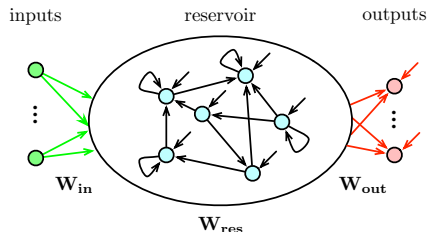
Echo State Networks (ESNs) are specific kinds of recurrent neural networks. An ESN consists of:

- An *input layer*
- A *reservoir* of neurons: random, recurrent and sparse
- An *output layer*
- ★ Only the output weights are trained!



ESNs: DYNAMICS

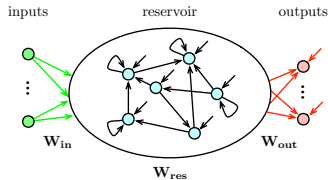
- We consider **leaky integrator ESNs (LI-ESNs)**:



$$\begin{aligned}\tilde{\mathbf{x}}(t+1) &= f_{res}\left(\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}_{res}\mathbf{x}(t)\right) \\ \mathbf{x}(t+1) &= (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1) \quad \alpha \text{ leaking rate} \\ \mathbf{y}(t+1) &= f_{out}\left(\mathbf{W}_{out}\mathbf{x}(t+1)\right)\end{aligned}$$

ESNs: INITIALIZATION

- Input weights $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_x \times (1+N_u)}$ (fixed):
Sampled from $\mathcal{U}(-a, a)$, where a is the *input scaling*.
- Reservoir weights $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N_x \times N_x}$ (fixed):
Sampled from $\mathcal{U}(-1, 1)$, set to 0 with *sparsity rate* 0.99, and rescaled to have a specific *spectral radius*¹ $\rho < 1$.
- Output weights $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (1+N_x)}$ (trainable!):
Here, closed-form solution of a simple Ridge Regression.

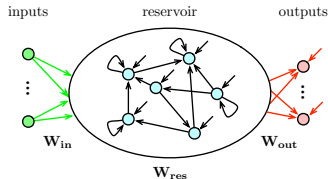


$$\begin{aligned}\tilde{\mathbf{x}}(t+1) &= f_{\text{res}}\left(\mathbf{W}_{\text{in}}\mathbf{u}(t+1) + \mathbf{W}_{\text{res}}\mathbf{x}(t)\right) \\ \mathbf{x}(t+1) &= (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1) \\ \mathbf{y}(t+1) &= f_{\text{out}}\left(\mathbf{W}_{\text{out}}\mathbf{x}(t+1)\right)\end{aligned}$$

¹The largest absolute value of the eigenvalues of \mathbf{W}_{res} .

ESNs: INITIALIZATION

- Input weights $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_x \times (1+N_u)}$ (fixed):
Sampled from $\mathcal{U}(-a, a)$, where a is the *input scaling*.
- Reservoir weights $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N_x \times N_x}$ (fixed):
Sampled from $\mathcal{U}(-1, 1)$, set to 0 with *sparsity rate* 0.99, and rescaled to have a specific *spectral radius*¹ $\rho < 1$.
- Output weights $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (1+N_x)}$ (trainable!):
Here, closed-form solution of a simple Ridge Regression.

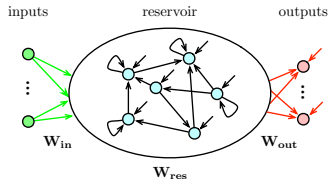


$$\begin{aligned}\tilde{\mathbf{x}}(t+1) &= f_{\text{res}}\left(\mathbf{W}_{\text{in}}\mathbf{u}(t+1) + \mathbf{W}_{\text{res}}\mathbf{x}(t)\right) \\ \mathbf{x}(t+1) &= (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1) \\ \mathbf{y}(t+1) &= f_{\text{out}}\left(\mathbf{W}_{\text{out}}\mathbf{x}(t+1)\right)\end{aligned}$$

¹The largest absolute value of the eigenvalues of \mathbf{W}_{res} .

ESNs: INITIALIZATION

- Input weights $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_x \times (1+N_u)}$ (fixed):
Sampled from $\mathcal{U}(-a, a)$, where a is the *input scaling*.
- Reservoir weights $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N_x \times N_x}$ (fixed):
Sampled from $\mathcal{U}(-1, 1)$, set to 0 with *sparsity rate* 0.99, and rescaled to have a specific *spectral radius*¹ $\rho < 1$.
- Output weights $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_y \times (1+N_x)}$ (trainable!):
Here, closed-form solution of a simple Ridge Regression.

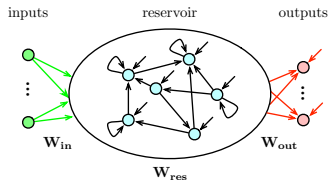


$$\begin{aligned}\tilde{\mathbf{x}}(t+1) &= f_{\text{res}}\left(\mathbf{W}_{\text{in}}\mathbf{u}(t+1) + \mathbf{W}_{\text{res}}\mathbf{x}(t)\right) \\ \mathbf{x}(t+1) &= (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1) \\ \mathbf{y}(t+1) &= f_{\text{out}}\left(\mathbf{W}_{\text{out}}\mathbf{x}(t+1)\right)\end{aligned}$$

¹The largest absolute value of the eigenvalues of \mathbf{W}_{res} .

ESNs: HYPERPARAMETERS

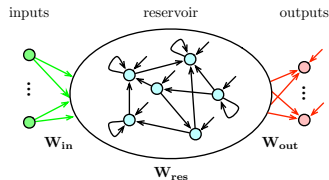
- *Input scaling α* : influences the nonlinearity of the reservoir:
Larger input scalings drive the reservoir units into larger activation values, hence into more non-linear regimes.
- *Spectral Radius ρ* : modulates the effect of past inputs:
Larger spectral radii correspond to longer input memories.
- *Leaking rate α* : controls the speed of the reservoir updating.
Larger leaking rates generate faster reacting reservoirs (reduced contribution of previous state in the updated state).



$$\begin{aligned}\tilde{\mathbf{x}}(t+1) &= f_{res}\left(\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}_{res}\mathbf{x}(t)\right) \\ \mathbf{x}(t+1) &= (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1) \\ \mathbf{y}(t+1) &= f_{out}\left(\mathbf{W}_{out}\mathbf{x}(t+1)\right)\end{aligned}$$

ESNs: HYPERPARAMETERS

- *Input scaling α* : influences the nonlinearity of the reservoir:
Larger input scalings drive the reservoir units into larger activation values, hence into more non-linear regimes.
- *Spectral Radius ρ* : modulates the effect of past inputs:
Larger spectral radii correspond to longer input memories.
- *Leaking rate α* : controls the speed of the reservoir updating.
Larger leaking rates generate faster reacting reservoirs (reduced contribution of previous state in the updated state).



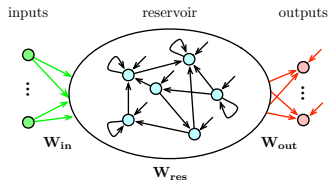
$$\tilde{\mathbf{x}}(t+1) = f_{res}(\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}_{res}\mathbf{x}(t))$$

$$\mathbf{x}(t+1) = (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1)$$

$$\mathbf{y}(t+1) = f_{out}(\mathbf{W}_{out}\mathbf{x}(t+1))$$

ESNs: HYPERPARAMETERS

- *Input scaling* α : influences the nonlinearity of the reservoir:
Larger input scalings drive the reservoir units into larger activation values, hence into more non-linear regimes.
- *Spectral Radius* ρ : modulates the effect of past inputs:
Larger spectral radii correspond to longer input memories.
- *Leaking rate* α : controls the speed of the reservoir updating.
Larger leaking rates generate faster reacting reservoirs (reduced contribution of previous state in the updated state).



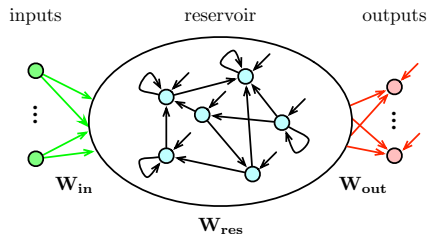
$$\tilde{\mathbf{x}}(t+1) = f_{res}(\mathbf{W}_{in}\mathbf{u}(t+1) + \mathbf{W}_{res}\mathbf{x}(t))$$

$$\mathbf{x}(t+1) = (1 - \alpha)\mathbf{x}(t) + \alpha\tilde{\mathbf{x}}(t+1)$$

$$\mathbf{y}(t+1) = f_{out}(\mathbf{W}_{out}\mathbf{x}(t+1))$$

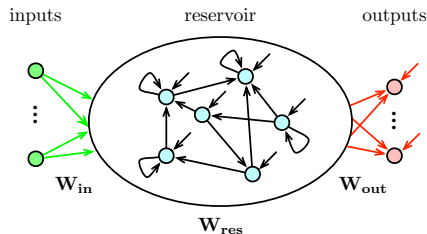
ESNs: INPUT FEATURES

- We consider the pre-trained dynamic word embedding BERT as input features.
- BERT-base: 12 encoder layers of the transformer model.
- Any input text is embedded into a sequence of 768-dimensional vectors. Other embeddings can be considered: FastText (300d), GloVe (300d), etc.



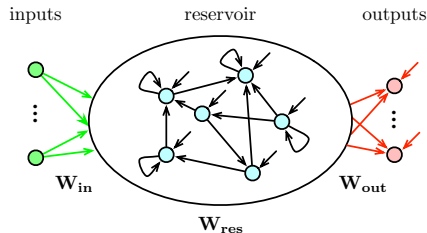
ESNs: INPUT FEATURES

- We consider the pre-trained dynamic word embedding BERT as input features.
- BERT-base: 12 encoder layers of the transformer model.
- Any input text is embedded into a sequence of 768-dimensional vectors. Other embeddings can be considered: FastText (300d), GloVe (300d), etc.



ESNs: INPUT FEATURES

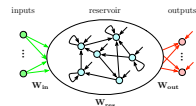
- We consider the pre-trained dynamic word embedding BERT as input features.
- BERT-base: 12 encoder layers of the transformer model.
- Any input text is embedded into a sequence of 768-dimensional vectors. Other embeddings can be considered: FastText (300d), GloVe (300d), etc.



ESNs: TRAINING

Inputs
texts

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$



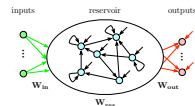
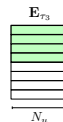
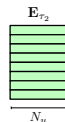
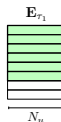
ESNs: TRAINING

Inputs
texts

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$

1. Embedding

Embedded
texts



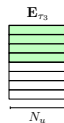
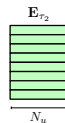
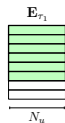
ESNs: TRAINING

Inputs
texts

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$

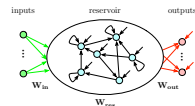
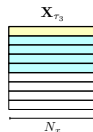
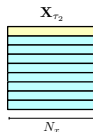
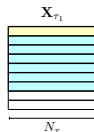
1. Embedding

Embedded
texts



2. ESN

Reservoir
states



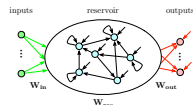
Inputs	$\tau_1 = (w_1^1, w_1^2, \dots)$	$\tau_2 = (w_2^1, w_2^2, \dots)$	$\tau_3 = (w_3^1, w_3^2, \dots)$	\dots
texts				

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$
 \mathbf{E}_{τ_1}

$$N_m$$
 \mathbf{X}_{τ_1}

$$\overline{N_T}$$
 \mathbf{x}_{T_1}

A horizontal bar with a double-headed arrow below it labeled N_T .

$$N_T$$


ESNs: TRAINING

Inputs
texts

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$

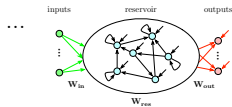
1. Embedding

Embedded
texts

 E_{τ_1}

 N_u
 E_{τ_2}

 N_u
 E_{τ_3}

 N_u


2. ESN

Reservoir
states

 X_{τ_1}

 N_x
 X_{τ_2}

 N_x
 X_{τ_3}

 N_x

3. Pooling

Merged
states

 x_{τ_1}

 N_x
 x_{τ_2}

 N_x
 x_{τ_3}

 N_x

Labels

 y_1^{label}

 N_y
 y_2^{label}

 N_y
 y_3^{label}

 N_y

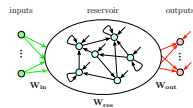
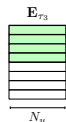
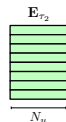
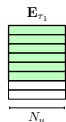
ESNs: TRAINING

Inputs
texts

$$\tau_1 = (w_1^1, w_1^2, \dots) \quad \tau_2 = (w_2^1, w_2^2, \dots) \quad \tau_3 = (w_3^1, w_3^2, \dots) \quad \dots$$

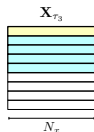
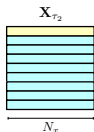
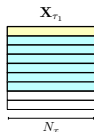
1. Embedding

Embedded
texts



2. ESN

Reservoir
states



3. Pooling

Merged
states



Labels



4. Learning

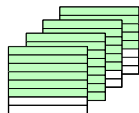


Ridge regression
(closed-form solution)

$$\hat{\mathbf{B}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{Y}^{\text{label}}$$

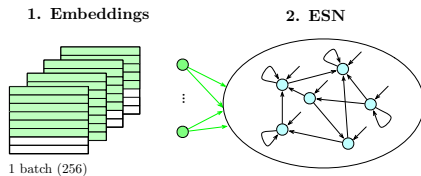
ESNs: TRAINING (BATCH PARALLELIZATION)

1. Embeddings

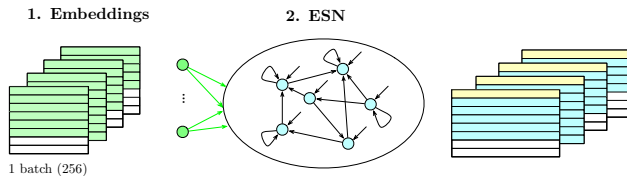


1 batch (256)

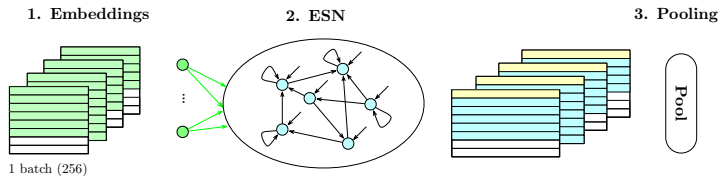
ESNs: TRAINING (BATCH PARALLELIZATION)



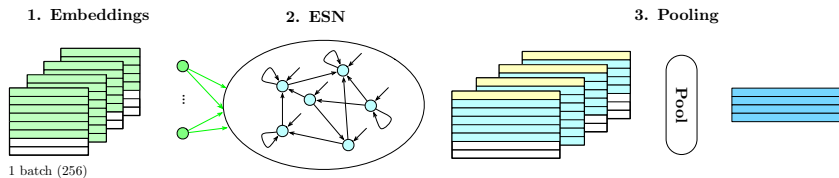
ESNs: TRAINING (BATCH PARALLELIZATION)



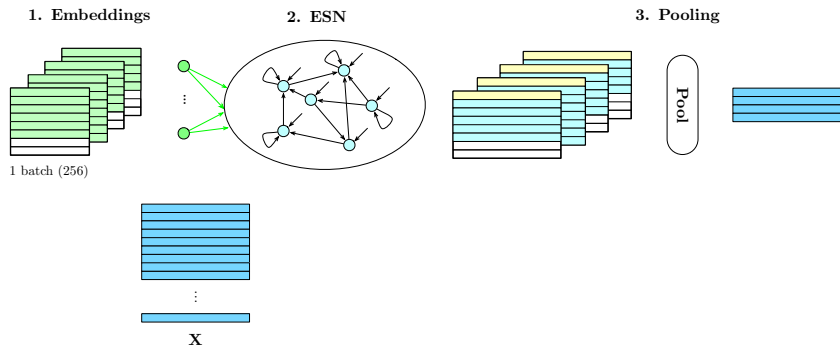
ESNs: TRAINING (BATCH PARALLELIZATION)



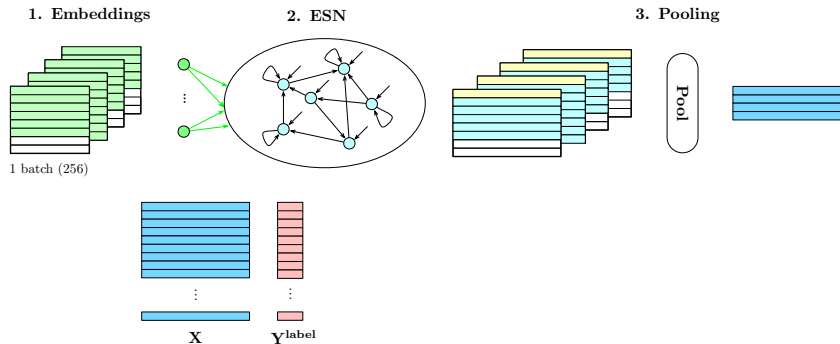
ESNs: TRAINING (BATCH PARALLELIZATION)



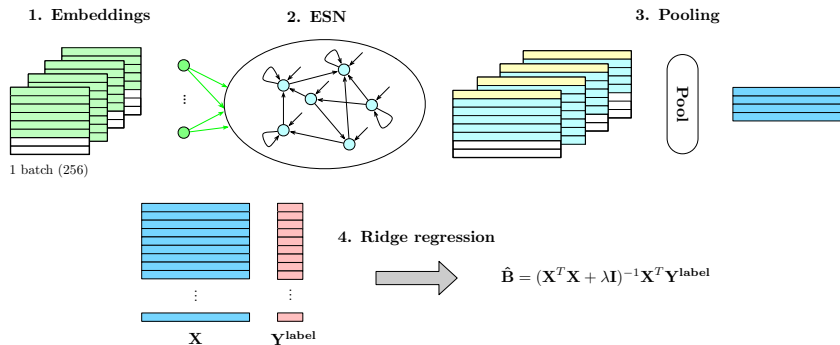
ESNs: TRAINING (BATCH PARALLELIZATION)



ESNs: TRAINING (BATCH PARALLELIZATION)



ESNs: TRAINING (BATCH PARALLELIZATION)



BERT EMBEDDING

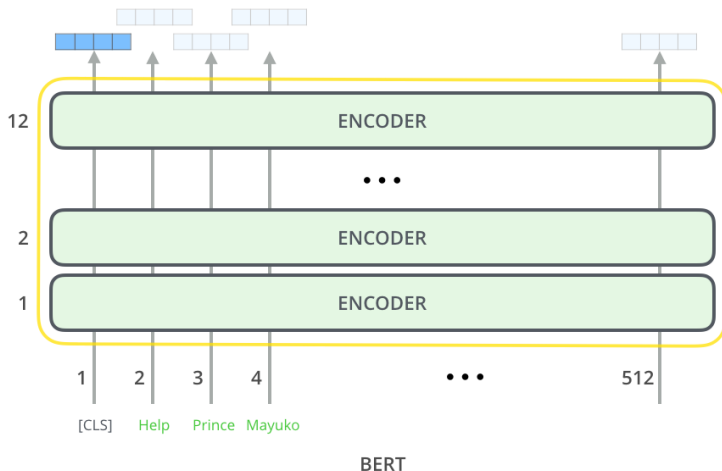


Figure: Jay Alamar's blog: <https://jalamar.github.io/illustrated-bert/>

Bi-LSTM

$\mathbf{u}(0)$

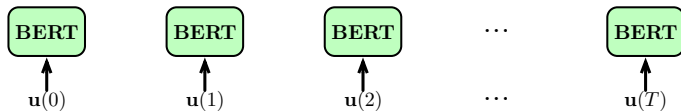
$\mathbf{u}(1)$

$\mathbf{u}(2)$

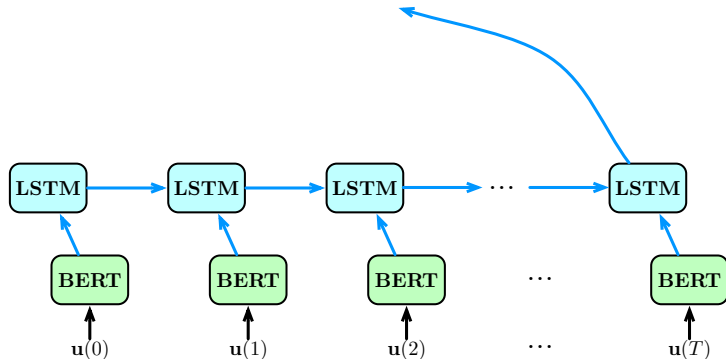
...

$\mathbf{u}(T)$

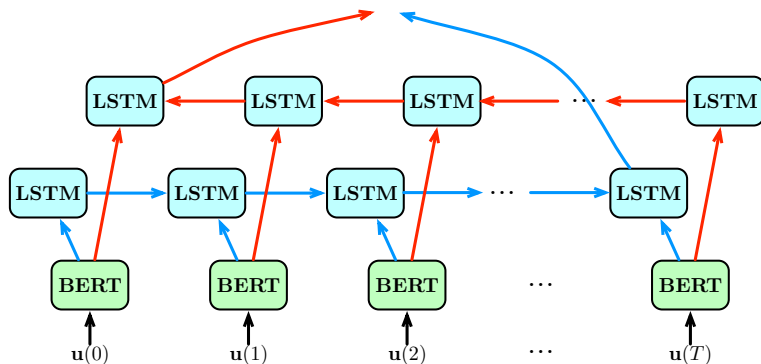
Bi-LSTM



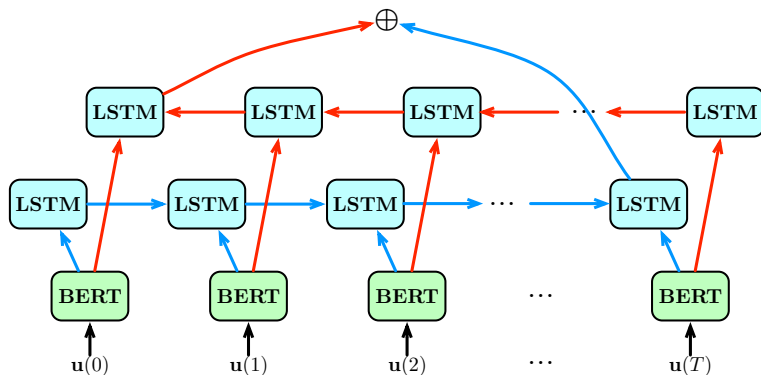
Bi-LSTM



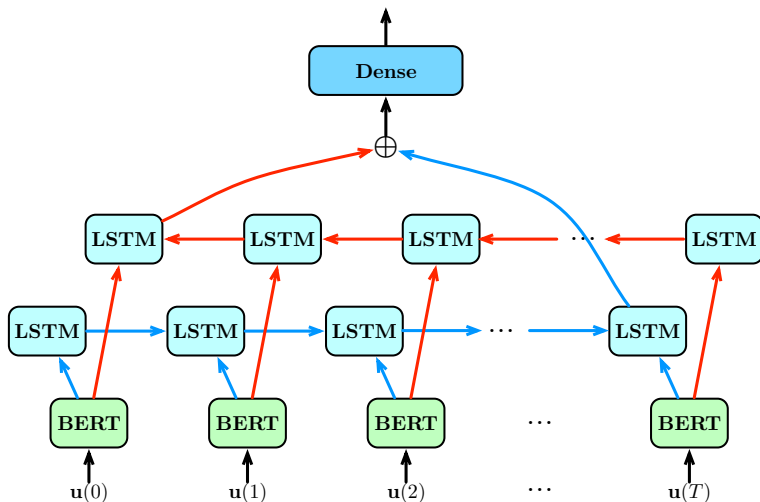
Bi-LSTM



Bi-LSTM



Bi-LSTM



BERT FINE-TUNED

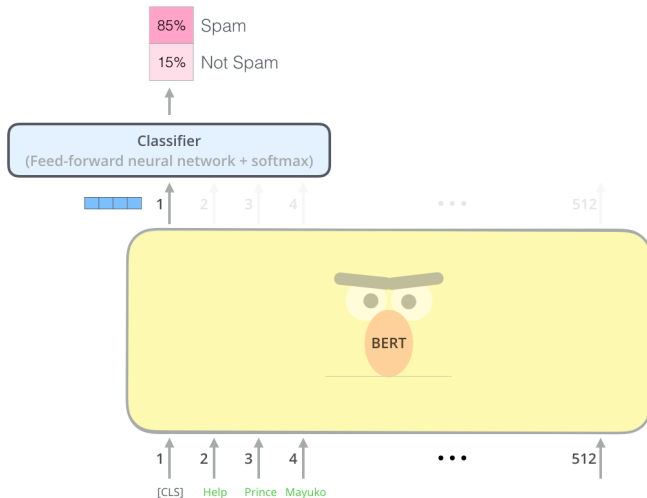


Figure: Jay Alamr's blog: <https://jalammar.github.io/illustrated-bert/>

RESULTS: EFFECT OF EMBEDDING

- The quality of the pre-trained embedding plays an important role (might be counter intuitive).

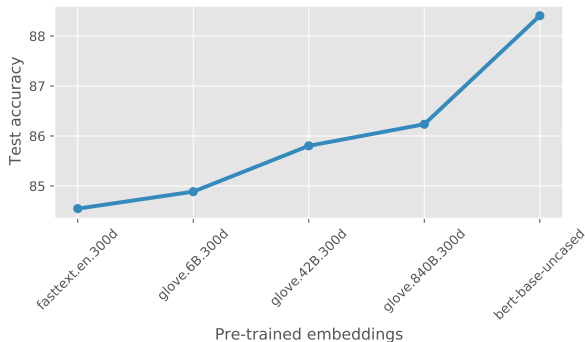


Figure: Test accuracy (%) on the IMDB dataset. ESN with *mean* pooling and reservoir size 1000 employing 5 different pre-trained word embeddings as features. Results are averaged over 5 random seeds.

RESULTS: EFFECT OF EMBEDDING

- ▶ The representational properties of the BERT embedding are not destroyed by the random input projection \mathbf{W}_{in} .
- Take $N = 750$ BERT-embedded words: $\mathbf{e}_1, \dots, \mathbf{e}_N$
- Pairwise cosine distances before and after \mathbf{W}_{in} projection:

$$d_{ij} := d_{\cos}(\mathbf{e}_i, \mathbf{e}_j) \quad \text{and} \quad d'_{ij} := d_{\cos}(\mathbf{W}_{\text{in}}\mathbf{e}_i, \mathbf{W}_{\text{in}}\mathbf{e}_j)$$

- Mean squared error between the distances d_{ij} and d'_{ij} (represents the distortion of the cosine distances induced by \mathbf{W}_{in})

$$\text{MSE}_{d,d'} := \frac{1}{\binom{N}{2}} \sum_{\{(i,j): i < j\}} (d_{ij} - d'_{ij})^2.$$

RESULTS: EFFECT OF EMBEDDING

- ▶ The representational properties of the BERT embedding are not destroyed by the random input projection \mathbf{W}_{in} .
- Take $N = 750$ BERT-embedded words: $\mathbf{e}_1, \dots, \mathbf{e}_N$
- Pairwise cosine distances before and after \mathbf{W}_{in} projection:

$$d_{ij} := d_{\cos}(\mathbf{e}_i, \mathbf{e}_j) \quad \text{and} \quad d'_{ij} := d_{\cos}(\mathbf{W}_{\text{in}}\mathbf{e}_i, \mathbf{W}_{\text{in}}\mathbf{e}_j)$$

- Mean squared error between the distances d_{ij} and d'_{ij} (represents the distortion of the cosine distances induced by \mathbf{W}_{in})

$$\text{MSE}_{d,d'} := \frac{1}{\binom{N}{2}} \sum_{\{(i,j): i < j\}} (d_{ij} - d'_{ij})^2.$$

RESULTS: EFFECT OF EMBEDDING

- ▶ The representational properties of the BERT embedding are not destroyed by the random input projection \mathbf{W}_{in} .
- Take $N = 750$ BERT-embedded words: $\mathbf{e}_1, \dots, \mathbf{e}_N$
- Pairwise cosine distances before and after \mathbf{W}_{in} projection:

$$d_{ij} := d_{\cos}(\mathbf{e}_i, \mathbf{e}_j) \quad \text{and} \quad d'_{ij} := d_{\cos}(\mathbf{W}_{\text{in}}\mathbf{e}_i, \mathbf{W}_{\text{in}}\mathbf{e}_j)$$

- Mean squared error between the distances d_{ij} and d'_{ij} (represents the distortion of the cosine distances induced by \mathbf{W}_{in})

$$\text{MSE}_{d,d'} := \frac{1}{\binom{N}{2}} \sum_{\{(i,j): i < j\}} (d_{ij} - d'_{ij})^2.$$

RESULTS: EFFECT OF EMBEDDING

- ▶ The representational properties of the BERT embedding are not destroyed by the random input projection \mathbf{W}_{in} .
- Take $N = 750$ BERT-embedded words: $\mathbf{e}_1, \dots, \mathbf{e}_N$
- Pairwise cosine distances before and after \mathbf{W}_{in} projection:

$$d_{ij} := d_{\cos}(\mathbf{e}_i, \mathbf{e}_j) \quad \text{and} \quad d'_{ij} := d_{\cos}(\mathbf{W}_{\text{in}}\mathbf{e}_i, \mathbf{W}_{\text{in}}\mathbf{e}_j)$$

- Mean squared error between the distances d_{ij} and d'_{ij} (represents the distortion of the cosine distances induced by \mathbf{W}_{in})

$$\text{MSE}_{d,d'} := \frac{1}{\binom{N}{2}} \sum_{\{(i,j): i < j\}} (d_{ij} - d'_{ij})^2.$$

RESULTS: EFFECT OF EMBEDDING

- The representational properties of the BERT embedding are not destroyed by the random input projection \mathbf{W}_{in} .

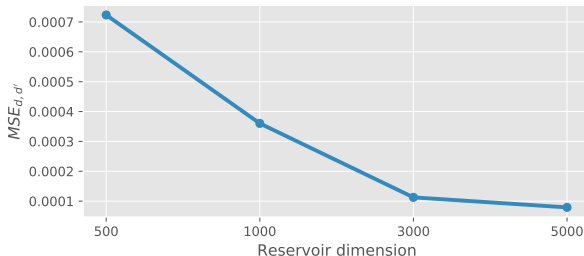


Figure: $MSE_{d,d'}$ as a function of the reservoir dimension N_x of $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_x \times 768}$. The matrices \mathbf{W}_{in} are the input weights of ESNs of sizes $N_x \in \{500, 1000, 3000, 5000\}$ whose input scalings have been optimized by BO on the IMDB dataset. Results are averaged over 5 random seeds. Larger input projections lead to smaller $MSE_{d,d'}$ values.

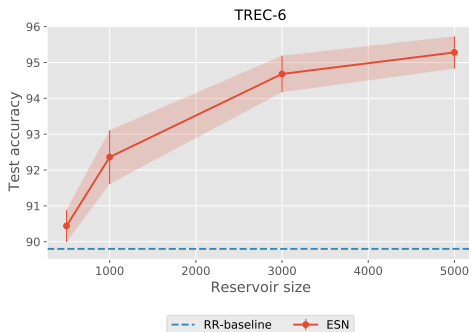
RESULTS: EFFECT OF THE RESERVOIR

- The temporal dynamics captured by the reservoir significantly improves the results.

• ESN = EMB + RES + POOL + RIDGE

• RR-baseline = EMB + + POOL + RIDGE

⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



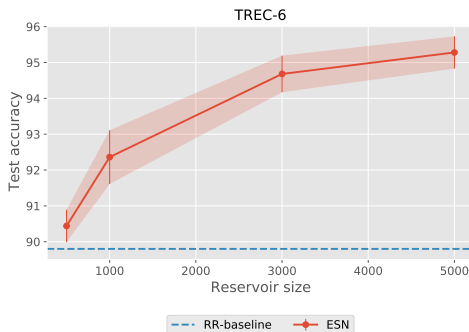
RESULTS: EFFECT OF THE RESERVOIR

- The temporal dynamics captured by the reservoir significantly improves the results.

● **ESN** = EMB + RES + POOL + RIDGE

● **RR-baseline** = EMB + + POOL + RIDGE

⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.

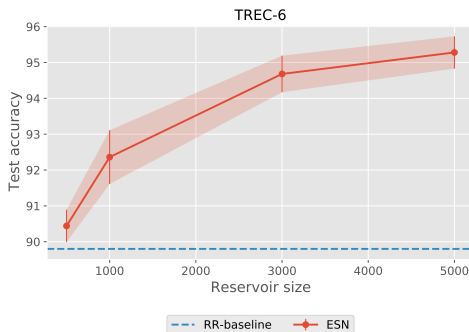


RESULTS: EFFECT OF THE RESERVOIR

- The temporal dynamics captured by the reservoir significantly improves the results.

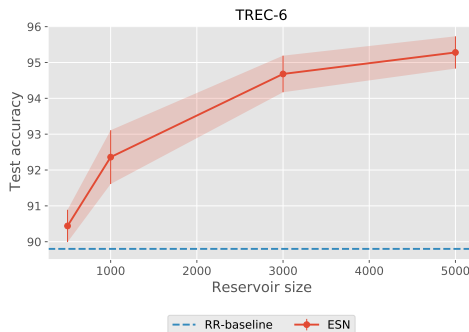
- **ESN** = EMB + RES + POOL + RIDGE
- **RR-baseline** = EMB + + POOL + RIDGE

⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



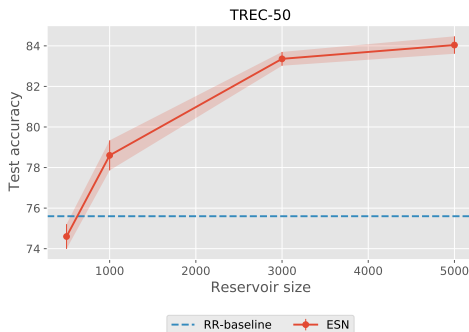
RESULTS: EFFECT OF THE RESERVOIR

- ▶ The temporal dynamics captured by the reservoir significantly improves the results.
 - **ESN** = EMB + RES + POOL + RIDGE
 - **RR-baseline** = EMB + + POOL + RIDGE
- ⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



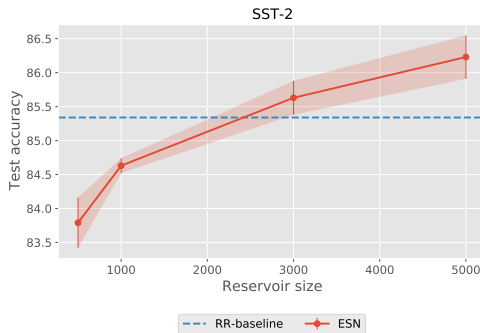
RESULTS: EFFECT OF THE RESERVOIR

- ▶ The temporal dynamics captured by the reservoir significantly improves the results.
 - **ESN** = EMB + RES + POOL + RIDGE
 - **RR-baseline** = EMB + + POOL + RIDGE
- ⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



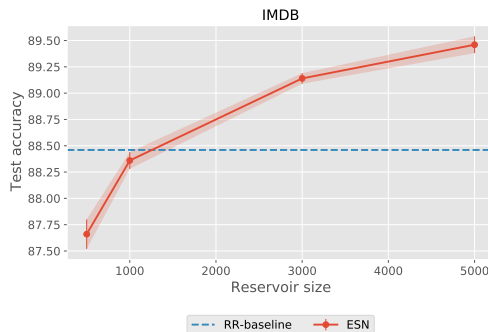
RESULTS: EFFECT OF THE RESERVOIR

- ▶ The temporal dynamics captured by the reservoir significantly improves the results.
 - **ESN** = EMB + RES + POOL + RIDGE
 - **RR-baseline** = EMB + + POOL + RIDGE
- ⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



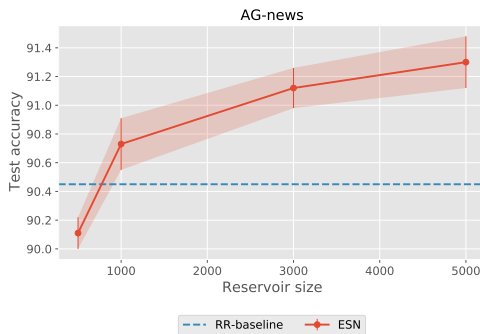
RESULTS: EFFECT OF THE RESERVOIR

- ▶ The temporal dynamics captured by the reservoir significantly improves the results.
 - **ESN** = EMB + RES + POOL + RIDGE
 - **RR-baseline** = EMB + + POOL + RIDGE
- ⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



RESULTS: EFFECT OF THE RESERVOIR

- ▶ The temporal dynamics captured by the reservoir significantly improves the results.
 - **ESN** = EMB + RES + POOL + RIDGE
 - **RR-baseline** = EMB + + POOL + RIDGE
- ⇒ ESN vs RR-baseline allows to assess the proper contribution of the reservoir.



RESULTS: COMPARISON WITH BI-LSTM AND BERT FINE-TUNED

- Overall, ESNs achieve competitive results with significantly faster training times.²

	TREC-6	TREC-50	SST-2	IMDb	AG News
RR-baseline	89.80	75.60	85.34	88.46	90.45
ESN-500	90.44 \pm 0.45	74.60 \pm 0.61	83.79 \pm 0.37	87.66 \pm 0.14	90.11 \pm 0.11
ESN-1000	92.36 \pm 0.75	78.60 \pm 0.74	84.63 \pm 0.11	88.36 \pm 0.08	90.73 \pm 0.18
ESN-3000	94.68 \pm 0.51	83.36 \pm 0.34	85.63 \pm 0.25	89.14 \pm 0.05	91.12 \pm 0.14
ESN-5000	95.28 \pm 0.45	84.04 \pm 0.43	86.23 \pm 0.32	89.46 \pm 0.08	91.30 \pm 0.18
Bi-LSTM	92.95	77.81	86.27	91.77	93.18
BERT-ft	96.60	79.40	92.20	93.60	94.24
BERT-ft*	97.20	—	93.50	94.60 / 95.63	94.75

Table: Test accuracy, in percentage (%). The ESN results are averaged over 5 random seeds. The Bi-LSTM networks have 2 hidden layers of 128 units. BERT-ft (our results) and BERT-ft* (literature) denote the BERT model downloaded as pre-trained and then fine-tuned on the datasets.

²1 GPU, 32 GB, NVIDIA V100

RESULTS: COMPARISON WITH BI-LSTM AND BERT FINE-TUNED

- Overall, ESNs achieve competitive results with significantly faster training times.³

	TREC-6	TREC-50	SST-2	IMDb	AG News
RR-baseline	7.32	8.16	105.52	97.09	153.58
ESN-500	9.47 ± 0.11	13.39 ± 0.54	115.10 ± 1.41	163.73 ± 0.47	297.90 ± 3.33
ESN-1000	9.63 ± 0.14	13.81 ± 0.36	115.42 ± 0.96	165.48 ± 0.43	297.37 ± 4.82
ESN-3000	10.68 ± 0.15	14.88 ± 0.23	116.39 ± 0.84	174.91 ± 0.16	298.79 ± 4.95
ESN-5000	16.24 ± 0.38	19.91 ± 0.46	121.94 ± 1.37	192.48 ± 0.30	304.08 ± 5.17
Bi-LSTM	97.02	95.92	1201.88	1630.05	2855.07
BERT-ft	556.38	549.17	4065.53	6235.73	64566.63
BERT-ft*	—	—	—	—	—

Table: Training time, in seconds (s.), of the different models over the five datasets TREC-6, TREC-50, SST-2, IMDb and AG News. The ESN results are averaged over 5 random seeds. The Bi-LSTM networks have 2 hidden layers of 128 units. BERT-ft (our results) and BERT-ft* (literature) denote the BERT fine-tuned model.

³1 GPU, 32 GB, NVIDIA V100

RESULTS: COMPARISON WITH BI-LSTM AND BERT FINE-TUNED

- Overall, ESNs achieve competitive results with significantly faster training times.⁴

	TREC-6	TREC-50	SST-2	IMDb	AG News
Bi-LSTM (128)	6.0	4.8	9.9	8.5	9.4
BERT ft (ours)	34.3	27.6	33.3	32.4	212.3

Table: Ratios between the training times of the Bi-LSTM (128) or BERT fine-tuned models and the ESN-5000. The ESN-5000 networks are trained from 4.8 up to 9.9 times faster than Bi-LSTM (128), and from 27.6 up to 212.3 times faster than the BERT fine-tuned.

⁴1 GPU, 32 GB, NVIDIA V100

RESULTS: EFFECT OF EMBEDDING (AGAIN)

- The quality of the pre-trained embedding plays an important role.

	TREC-6 accuracy time	TREC-50 accuracy time	IMDb accuracy time
GloVe-ESNs	91.12 ± 0.48 2.64 ± 0.12	83.96 ± 0.23 7.02 ± 0.14	87.78 ± 0.11 38.47 ± 0.14
BERT-ESNs	95.28 ± 0.45 16.24 ± 0.38	84.04 ± 0.43 19.91 ± 0.46	89.46 ± 0.08 192.48 ± 0.30

Table: Test accuracy (%) and training time (sec.) of the GloVe-feature and BERT-feature ESNs on the TREC-6, TREC-50 and IMDb datasets. The BERT-feature ESNs achieve higher accuracies, but with higher training times also.

CONCLUSIONS

- Hyperparameter tuning via Bayesian optimization (BO) takes time (even if significantly faster than grid search).
- The consideration of bi-directional ESNs does not significantly improve the results.
- ★ ESNs can be considered as robust, efficient and fast candidates for text classification tasks.
- ★ Transformer-based models like BERT achieve impressive performance but are very resource-consuming.
- ★ By contrast, our work fits within the context of light and fast-to-train models for NLP.

CONCLUSIONS

- Hyperparameter tuning via Bayesian optimization (BO) takes time (even if significantly faster than grid search).
- The consideration of bi-directional ESNs does not significantly improve the results.
- ★ ESNs can be considered as robust, efficient and fast candidates for text classification tasks.
- ★ Transformer-based models like BERT achieve impressive performance but are very resource-consuming.
- ★ By contrast, our work fits within the context of light and fast-to-train models for NLP.

CONCLUSIONS

- Hyperparameter tuning via Bayesian optimization (BO) takes time (even if significantly faster than grid search).
- The consideration of bi-directional ESNs does not significantly improve the results.
- ★ ESNs can be considered as robust, efficient and fast candidates for text classification tasks.
- ★ Transformer-based models like BERT achieve impressive performance but are very resource-consuming.
- ★ By contrast, our work fits within the context of light and fast-to-train models for NLP.

CONCLUSIONS

- Hyperparameter tuning via Bayesian optimization (BO) takes time (even if significantly faster than grid search).
- The consideration of bi-directional ESNs does not significantly improve the results.
- ★ ESNs can be considered as robust, efficient and fast candidates for text classification tasks.
- ★ Transformer-based models like BERT achieve impressive performance but are very resource-consuming.
- ★ By contrast, our work fits within the context of light and fast-to-train models for NLP.

CONCLUSIONS

- Hyperparameter tuning via Bayesian optimization (BO) takes time (even if significantly faster than grid search).
- The consideration of bi-directional ESNs does not significantly improve the results.
- ★ ESNs can be considered as robust, efficient and fast candidates for text classification tasks.
- ★ Transformer-based models like BERT achieve impressive performance but are very resource-consuming.
- ★ By contrast, our work fits within the context of light and fast-to-train models for NLP.

CONCLUSIONS

Thank you!