Tweet2Vec- Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder (Vosoughi et al., *SIGIR'16*)

Presented by - **Sidharth Singla**

INTRODUCTION

- **Tweet2Vec:** A novel method for generating general purpose vector representation of tweets.
- Character-level CNN-LSTM encoder-decoder model to learn tweet embeddings.
- Model trained on 3 million, randomly selected English language tweets.

MOTIVATION

- Extensive Feature Engineering: Creation of task-specific, hand-crafted features.
- Time consuming and inefficient: New features engineered for every task.
- **Tweet2Vec:** General purpose vector representation of tweets.
- Can be used with any classification model and for any classification task.

RELATED WORK

- Word Embeddings: Word2Vec; Encoder-Decoder using LSTM and GRU.
- Work by Le et al. “Distributed representations of sentences and documents”. Generated representations for sentences using Word2Vec model and called ParagraphVec.

INPUT REPRESENTATION

- Tweet representation: $X \in \{0, 1\}^{150 * 70}$
- 150 maximum number of characters in a tweet, including padding.
- 70 English characters in total. Includes English alphabets, numbers, special characters and unknown character.

MODEL ARCHITECTURE

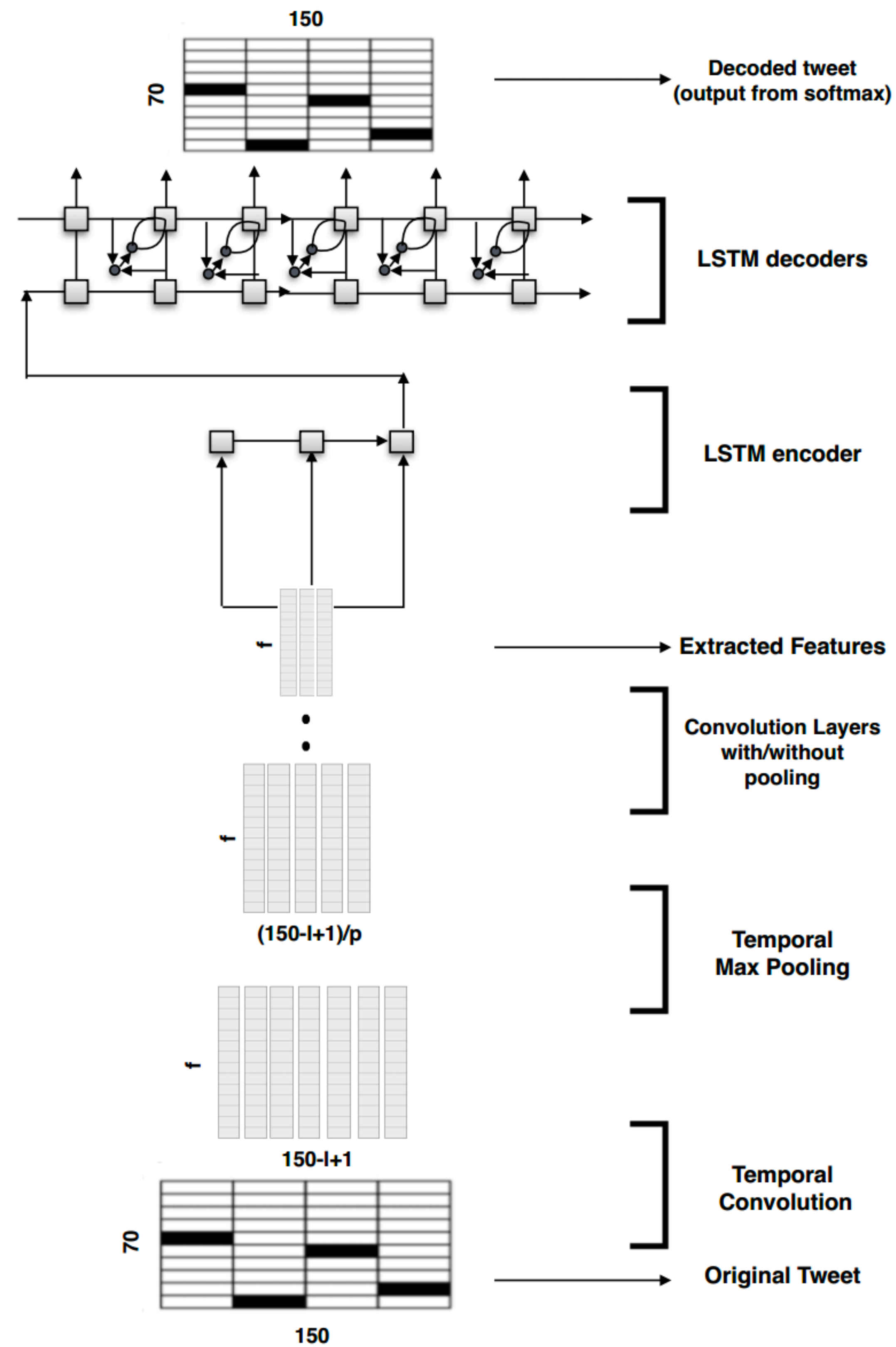
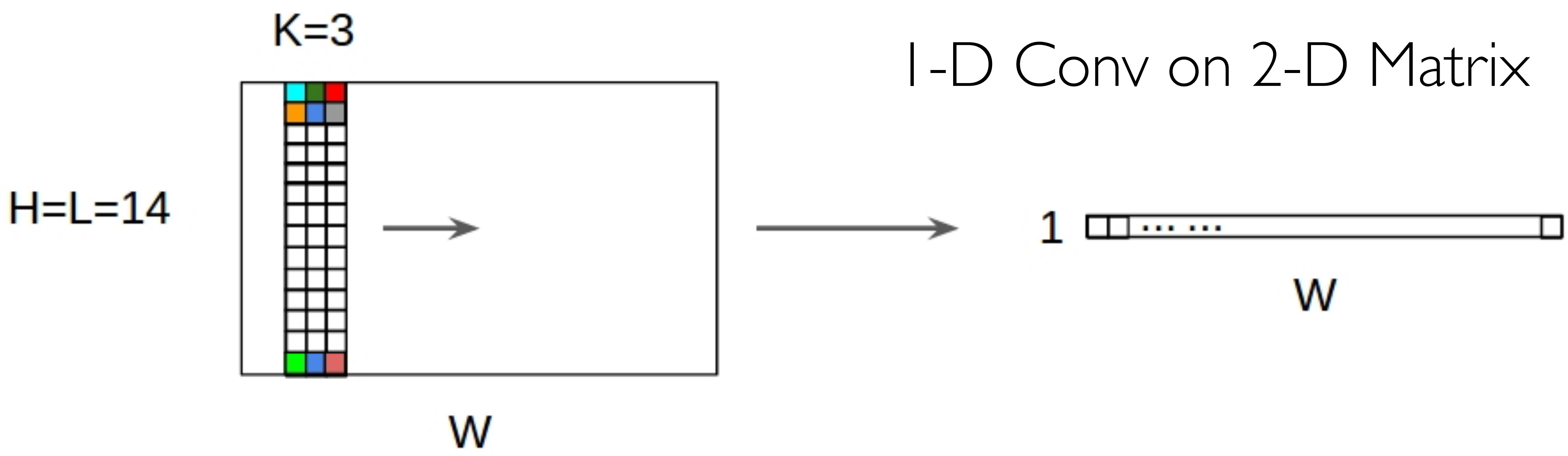
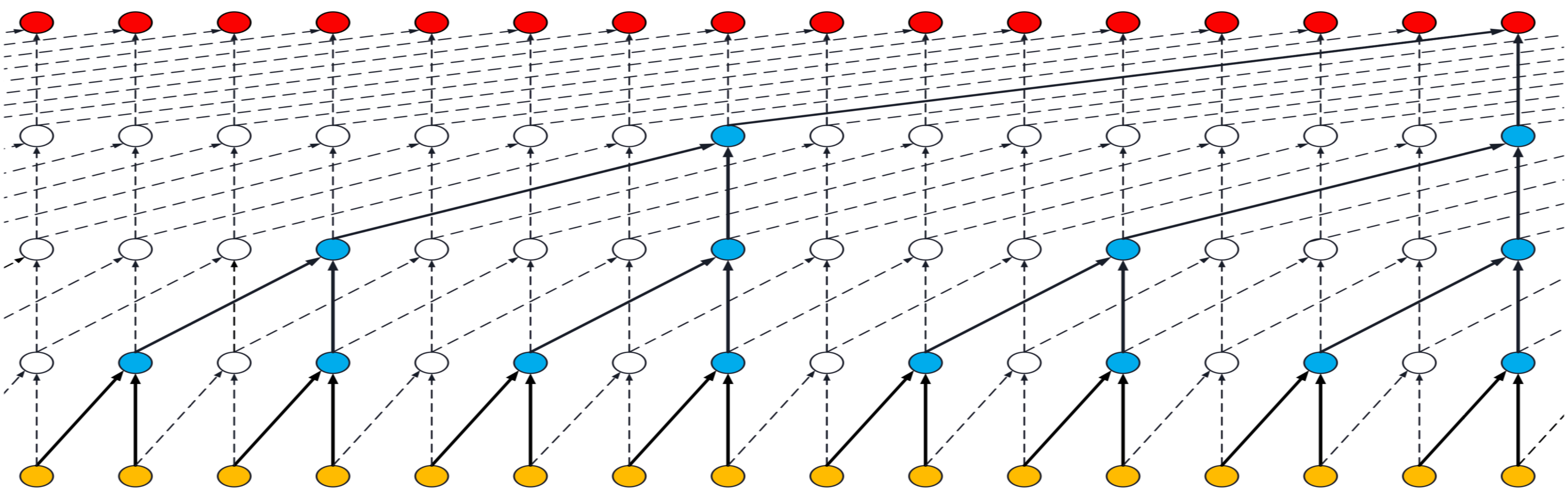


Figure 1: Illustration of the CNN-LSTM Encoder-Decoder Model

1-D Conv on 2-D Matrix

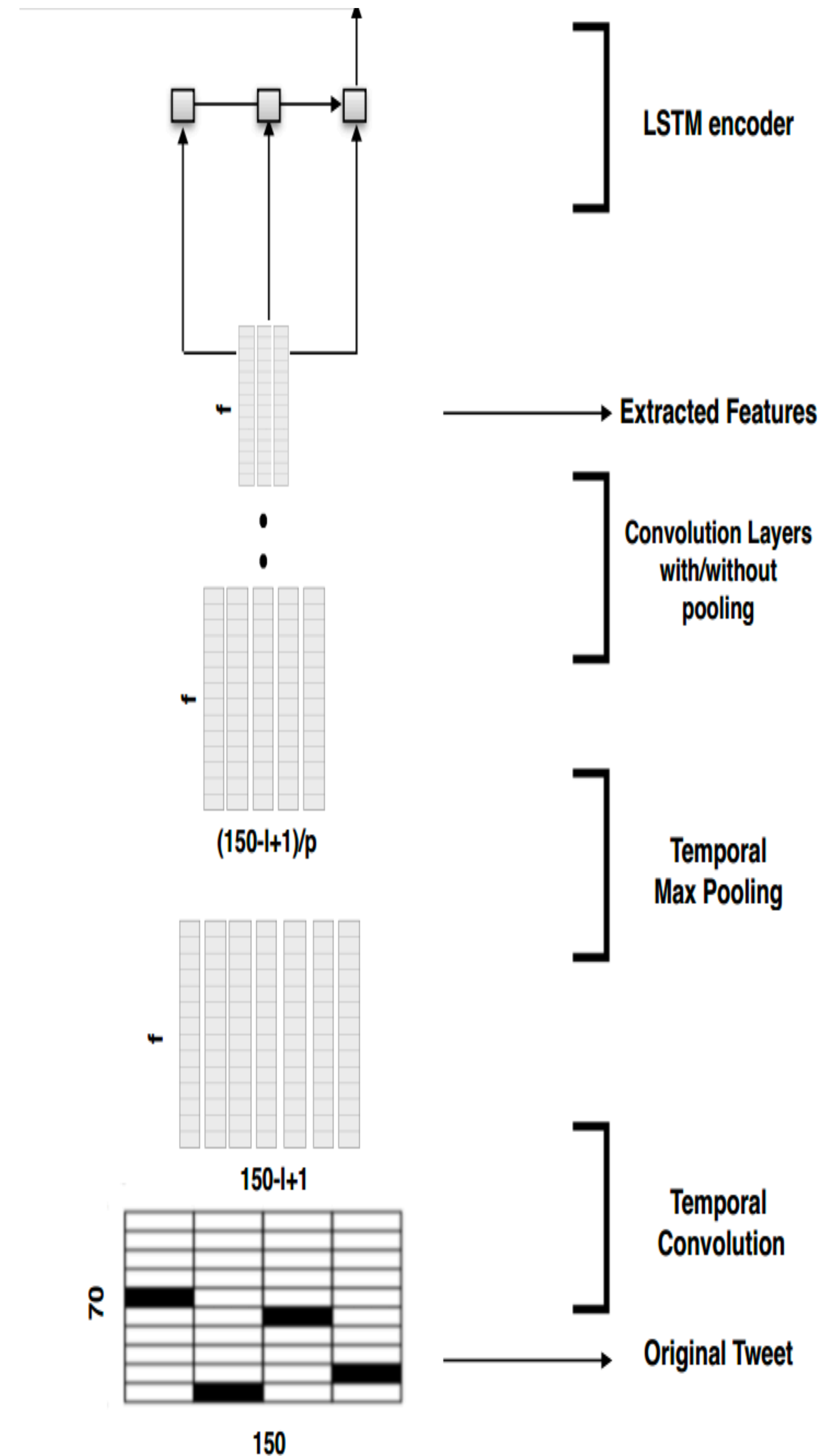


Temporal Conv



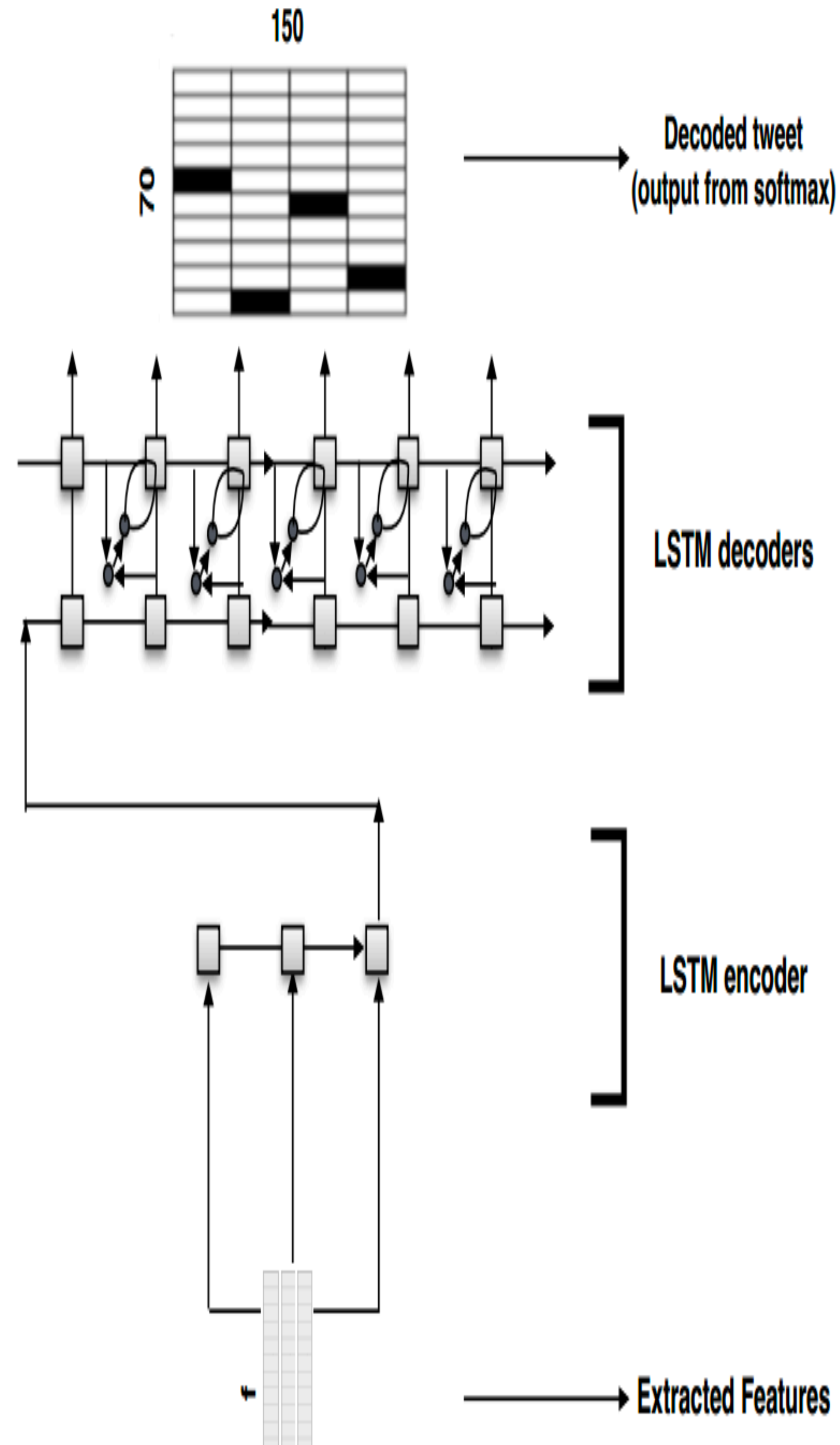
Convolutions are causal. No information leakage from future to past.

- Char-Level CNN Model:
  - Temporal 1-d convolutions(4 layers) and Temporal max-pooling.
  - Last conv layer output size: 10×512. Input to the LSTM layer.
  - No pooling at the higher layers.



- Encoder Model:
  - LSTM Model.
  - H_{conv} = CharCNN(T)
    $h_t = \mathrm{LSTM}(g_t; h_{t-1})$
    where $g = H_{conv}$
  - Final time-step vector output size: 256

- Decoder Model:
  - Two LSTM layers.
  - Last decoder generates each character C sequentially.
    $P(C_t | .) = \mathrm{softmax}(T_t, h_{t-1})$



DATA AUGMENTATION & LOSS

- Replicated tweets and replaced some of the words in the replicated tweets with the synonyms obtained from WordNet.
- In encoder-decoder model, encoded representation is decoded to the actual tweet or a synonym replaced version of the tweet from the augmented data.
- For regularization, a dropout mechanism is applied after the penultimate layer.
- Loss: Cross-Entropy Loss.

EXPERIMENTS

1. Tweet Semantic Relatedness

- Based on the SemEval 2015-Task 1: Paraphrase and Semantic Similarity in Twitter. Dataset contains 18K tweet pairs for training and 1K pairs for testing.
- Given two tweet vectors 'r' and 's', element-wise product 'r.s' and absolute difference $|r-s|$ is computed and concatenated.
- Logistic regression model is trained on these features using the dataset.

TWEET SEMANTIC RELATEDNESS RESULTS

Table 2: Results of the paraphrase and semantic similarity in Twitter task.

Model	*Precision*	*Recall*	$F1 - Score$
ParagraphVec	0.570	0.680	0.620
nnfeats	**0.767**	0.583	0.662
ikr	0.569	**0.806**	0.667
linearsvm	0.683	0.663	0.672
svckernel	0.680	0.669	0.674
Tweet2Vec	0.679	0.686	**0.677**

EXPERIMENTS

2. Tweet Sentiment Classification

- Based on the SemEval 2015-Task 10B: Twitter Message Polarity Classification. Classes are positive, negative or neutral in sentiment. The size of the training and test sets were 9,520 tweets and 2,380 tweets respectively.

- As with the last task, vector representation are extracted using Tweet2Vec and a logistic regression classifier is trained. Performance is measured as the average F1-score of the positive and the negative class.

TWEET SENTIMENT CLASSIFICATION RESULTS

Table 3: Results of Twitter sentiment classification task.

Model	*Precision*	*Recall*	*F1 − Score*
ParagraphVec	0.600	0.680	0.637
INESC-ID	N/A	N/A	0.642
lsislif	N/A	N/A	0.643
unitn	N/A	N/A	0.646
Webis	N/A	N/A	0.648
Tweet2Vec	0.675	0.719	**0.656**

CONCLUSION

- Generic embeddings for Twitter based classification tasks.
- Can be used with any classification model.
- Can be used to cluster tweets based on similarity.

FUTURE WORK

1. Data Augmentation through reordering the words in the tweets to make the model robust to word-order.

2. Attention mechanism exploitation in the model to improve alignment of words in tweets during decoding.