Data-efficient Alignment of Multimodal Sequences by Aligning Gradient Updates and Internal Feature Distributions

Jianan Wang[1], Boyang Li[2,3], Xiangyu Fan[1], Jing Lin[1], and Yanwei Fu[1,4]

[1]School of Data Science, Fudan University, China
[2]Nanyang Technological University, Singapore
[3]Alibaba-NTU Singapore Joint Research Institute
[4]MOE Frontiers Center for Brain Science, Fudan University, China

boyangli@outlook.com   jinglin0224@163.com   {jawang19, fanxy19, yanweifu}@fudan.edu.cn

Abstract

The task of video and text sequence alignment is a prerequisite step toward joint understanding of movie videos and screenplays. However, supervised methods face the obstacle of limited realistic training data. With this paper, we attempt to enhance data efficiency of the end-to-end alignment network NeuMATCH [15]. Recent research [56] suggests that network components dealing with different modalities may overfit and generalize at different speeds, creating difficulties for training. We propose to employ (1) layer-wise adaptive rate scaling (LARS) to align the magnitudes of gradient updates in different layers and balance the pace of learning and (2) sequence-wise batch normalization (SBN) to align the internal feature distributions from different modalities. Finally, we leverage random projection to reduce the dimensionality of input features. On the YouTube Movie Summary dataset, the combined use of these technique closes the performance gap when the pretraining on the LSMDC dataset is omitted and achieves the state-of-the-art result. Extensive empirical comparisons and analysis reveal that these techniques improve optimization and regularize the network more effectively than two different setups of layer normalization.

1. Introduction

Today we have access to a massive amount of parallel video and textual data in the form of movie videos and screenplays. For instance, more than 1,200 movie scripts are available at the Internet Movie Script Database[1] under fair use. However, in their natural form, the data lack

---

[1]https://www.imsdb.com/

fine-grained correspondences at the level of sentences and video segments. Establishing such cross-modality correspondences will enable a variety of applications such as acquisition of multi-modal knowledge [51, 59, 46, 58, 63], generation of one modality from the other [41, 34, 54, 69, 57, 62, 30, 5, 33], movie video retrieval [37], and visualization of movie content [26]. For this purpose, we investigate the alignment video and textual sequences (see an example of aligned sequences in Figure 1).

A major obstacle of this task is the limited amount of training data due to the high cost of manually annotating sentence-level alignment. The Large Scale Movie Description Challenge (LSMDC) dataset [42] is massive (158 hours of video and 124k texts), but it lacks one-to-many matching, unmatched elements, and imprecise language that are common in real-world data. The YouTube Movie Summary (YMS) dataset [15] is more realistic but is more than 20 times smaller (6.7 hours of video and 5.9k texts).

For its small size, fast pace, complex correspondence relations, and storytelling language, the YMS dataset is a difficult challenge for training from scratch. In [15], the end-to-end sequence alignment network NeuMATCH is pretrained on LSMDC and finetuned on YMS. In our experiments, directly training NeuMATCH on YMS unsurprisingly results in overfitting and a 2.4% performance drop in text accuracy despite early stopping.

With this paper, we aim to enhance the data efficiency of video-text sequence alignment and mitigate overfitting when training on small data. Recent research [56] suggests that in multimodal neural networks, sub-networks dealing with different modalities overfit and generalize at different speeds. This is consistent with our observation that applying regularization uniformly to a network that already overfits can sometimes exacerbate overfitting. To overcome this

Figure 1: An example of aligned video and text sequences from YMS [15]. One text snippet can match multiple video clips and some video clips may be unmatched.

issue, we propose to (1) align the magnitudes of gradient-based parameter updates so that different modalities learn at similar rates and (2) align the internal feature distributions from different modalities.

To accomplish the first goal, we adopt layer-wise adaptive rate scaling (LARS), which is traditionally reserved for training with large batches [64, 65]. In our experiments, the optimization of the four network stacks are more balanced under LARS than Adam (see Section 4.4), suggesting LARS helps in properly pacing learning in different network components . This work may be considered as redirecting the existing technique of LARS to a novel objective. To accomplish the second goal, we adopt sequence-wise batch normalization (SBN) [28], which normalizes the internal feature distributions from different modalities. We show that SBN is superior to a popular configuration of layer normalization [4], which normalizes the internal representations of LSTMs instead of outputs.

Large pretrained video-text models (e.g., [48, 70, 35, 34]) can serve as powerful feature extractors for single video clips and single sentences. However, the NeuMATCH network contains video/text *sequence* encoders (a.k.a. the Video Stack and the Text Stack) that capture contextual information from long sequences. On average, they encode 180 clips and 66 text snippets for one video in YMS. It is not immediately clear how to apply existing methods to pretrain the sequence encoders. To handle potential overfitting in the sequence encoders, we propose to use random projection (RP) [1] to reduce the dimensionality of input features. After applying these techniques, we surpass the previous best performance with pretraining as well as a baseline with LXMERT [50] features, which is directly trained on YMS.

Contributions. First, we successfully improve the data efficiency of multimodal sequence alignment with three techniques, LARS, SBN, and RP. As a result, we achieve state-of-the-art results on the YMS dataset without pretraining on LSMDC. Second, by providing detailed empirical analysis of these techniques (Sections 4.4), 4.5), and 4.6)), we hint at their broader potential for improving data efficiency in multimodal tasks. To our knowledge, the combined use of the three techniques in multimodal tasks is novel.

2. Related Work

Alignment of Video and Text Sequences. The alignment of video and text sequences has long been a problem of research interest [13, 17]. A popular theme of solutions is metric learning followed by dynamic time warping, which can be understood as inference on a linear conditional random field (CRF) [44, 52, 53]. [7] adopts a constrained quadratic integer programming formulation. [71] is an early approach utilizing convolutional neural networks that consider a local context in learning similarity between video segments and book chapters. After that, a CRF approach is used for decoding. By decomposing the alignment into a sequence of action classification, NeuMATCH [15] provides the first end-to-end differentiable solution. However, due to the complexity and heterogeneity of the network, training can become a challenge.

To deal with data sparsity, researchers turn to weakly supervised or unsupervised learning. [9] utilizes the sequential constraints for weakly supervised action localization and proposes a differentiable dynamic time warping objective function. [66] discovers objects from textual descriptions of their locations and movements. [36] is an unsupervised approach for aligning textual instructions and demonstration videos based on object names and images.

Text Grounding in Video. Temporal grounding of text, also called moment retrieval or temporal activity localization, seeks a temporal interval of the video that corresponds to a textual query [68]. This is in contrast to spatio-temporal grounding, where texts are grounded in spatio-temporal tubes [11], and the spatial grounding (*e.g.*, [16, 55]), where texts are grounded in spatial locations of images. Here we briefly review fully supervised temporal grounding. TALL [18] compares the sentence to sliding-window clip proposals and tweaks their temporal locations. [22] learns to rank the correct proposal higher than incorrect ones. [61] uses a proposal network to predict the proposal's location directly. [10] formulates the task as sequence labeling, where the classification at each time step selects among proposals of predefined lengths ending at the current time. [21, 60] introduce reinforcement learning methods.

Temporal Activity Proposal. The proposal network, first used in object detection [39], is often used to identify potentially useful portions of video or image as the candidates for grounding. In the temporal grounding task, we conventionally place proposals of predefined lengths at all possible locations; proposals are classified as valid or invalid and their boundaries are adjusted [19]. The frame-based approach in [31] finds likely starting and ending positions of proposals and collects candidate proposals in-between. [32] creates proposals of different lengths using a downsampling and upsampling strategy similar to U-Net [43]. [25] introduces data augmentation techniques, time warping and time masking, in semi-supervised learning.

The semantic content of a natural language sentence incurs considerably greater variations than action labels, which usually contain 1-3 words. Thus, it is challenging to *a priori* determine good candidates for sentence grounding. For this reason, we let one sentence to match with multiple predefined video clips rather than assuming it matches with exactly one proposal found by a proposal network.

Pacing Multimodal Gradient Updates. End-to-end optimization has been a driving force for the many successes of deep learning, yet it is only understood recently that interactions between gradients for different network components may interfere with learning [29]. Wang et al. [56] employ a convex combination of unimodal gradients so that different modalities may learn at similar rates. However, the computation of unimodal gradients is not easily applicable in the sequence alignment task, which always requires input from two modalities. As an alternative, we propose to maintain the same ratio between the the gradient magnitudes and the layer weights, so that the paces of learning across network components may be aligned.

3. Approach

3.1. NeuMATCH

NeuMATCH [15] is an end-to-end differentiable neural network for multi-sequence alignment. The basic idea is to represent the current state of partially matched sequences as a vector and classify it into an action. Incrementally, the actions determine if the topmost elements in the two sequences should be matched or if they should be removed from consideration. After one decision, the second element may move to the top position and get processed. This process repeats until one of the sequences is exhausted.

Network Architecture. Shown in Figure 2, the NeuMATCH architecture employs four LSTM networks that respectively represent the video sequence, the text sequence, previously executed actions, and previously matched elements. The network only operates on the topmost elements on the four stacks.

The LSTM network [20] is commonly applied to sequential data and can be characterized as follows. Let the sequence of input features be $\langle \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T \rangle$, or $\langle \boldsymbol{x}_t \rangle_{t=1}^{T}$ for short. LSTM outputs a sequence of hidden states $\langle \boldsymbol{h}_t \rangle_{t=1}^{T}$ and internal cell states $\langle \boldsymbol{c}_t \rangle_{t=1}^{T}$. For any $t \in \{1 \ldots T\}$, $\boldsymbol{x}_t, \boldsymbol{h}_t, \boldsymbol{c}_t \in \mathcal{R}^D$. In temporally forward LSTM, $\boldsymbol{h}_t$ is updated as follows.

$$\left[\boldsymbol{i}_t^\top, \boldsymbol{f}_t^\top, \tilde{\boldsymbol{c}}_t^\top, \boldsymbol{o}_t^\top \right]^\top = W_x \boldsymbol{x}_t + W_h \boldsymbol{h}_{t-1} + \boldsymbol{b} \quad (1)$$

$$\boldsymbol{c}_t = \sigma(\boldsymbol{f}_t) \otimes \boldsymbol{c}_{t-1} + \sigma(\boldsymbol{i}_t) \otimes \tanh(\tilde{\boldsymbol{c}}_t) \quad (2)$$

$$\boldsymbol{h}_t = \sigma(\boldsymbol{o}_t) \otimes \tanh(\boldsymbol{c}_t) \quad (3)$$

where $W_x, W_h \in \mathcal{R}^{4D \times D}$ and $\boldsymbol{b} \in \mathcal{R}^{4D}$ are trainable parameters. $\otimes$ denotes component-wise multiplication. Note that LSTM may run temporally backward, so that $\boldsymbol{h}_t$ becomes a function of $\boldsymbol{x}_t, \boldsymbol{h}_{t+1}$, and $\boldsymbol{c}_{t+1}$, as in the Video and Text Stacks.

In the Video Stack and the Text Stack, the input sequence of video features is denoted as $\langle \boldsymbol{v}_i \rangle_{i=1}^{N}$ and the input text features are $\langle \boldsymbol{s}_j \rangle_{j=1}^{M}$. These features go through a random projection layer for dimensionality reduction (see Section 3.5) and fully connected (FC) layers before the LSTM. At any time t, the topmost element in the two stacks are denoted as $\boldsymbol{v}_t$ and $\boldsymbol{s}_t$ respectively. To model contextual information from the rest of the sequence, the two LSTMs feed temporally backward. The LSTM hidden states go through sequence-wise batch normalization (see Section 3.3), whose normalized output are $\langle \tilde{\boldsymbol{v}}_i \rangle_{i=1}^{N}$ and $\langle \tilde{\boldsymbol{s}}_j \rangle_{j=1}^{M}$.

The Action Stack, feeding temporally forward, provides contextual information from previous actions executed by the network. Its input sequence contains one-hot vectors $\langle \boldsymbol{a}_k \rangle_{k=1}^{t-1}$ and its output is the last hidden state $\tilde{\boldsymbol{a}}_{t-1}$. Similarly, the Matched Stack provides context from previously matched video and text elements. Its input sequence $\langle \boldsymbol{m}_k \rangle_{k=1}^{t-1}$ contains the concatenation of video and text features. For example, if at time k, the ith video and the jth text are matched, we let $\boldsymbol{m}_k = [\tilde{\boldsymbol{v}}_i; \tilde{\boldsymbol{s}}_j]$. In case of one-to-many matching, we take the average of multiple elements in the same modality and the same slot. The Matched Stack outputs the last hidden state, $\tilde{\boldsymbol{m}}_{t-1}$.

The network extracts an overall state vector $[\tilde{\boldsymbol{v}}_t; \tilde{\boldsymbol{s}}_t; \tilde{\boldsymbol{a}}_{t-1}; \tilde{\boldsymbol{m}}_{t-1}]$ from all four stacks. The state goes through two FC layers before a softmax function, where it is classified into one of the available actions as defined in the next paragraph. The network is trained using cross-entropy loss.

Action Definitions. We define five actions that can be used in the alignment of two sequences in a one-to-one or one-to-many setting with unmatched elements. The actions are Pop-Text, Pop-Video, Match, Match-Retain-Text, and Match-Retain-Video. Depending on the nature of the data, we may employ a subset of all possible actions. The
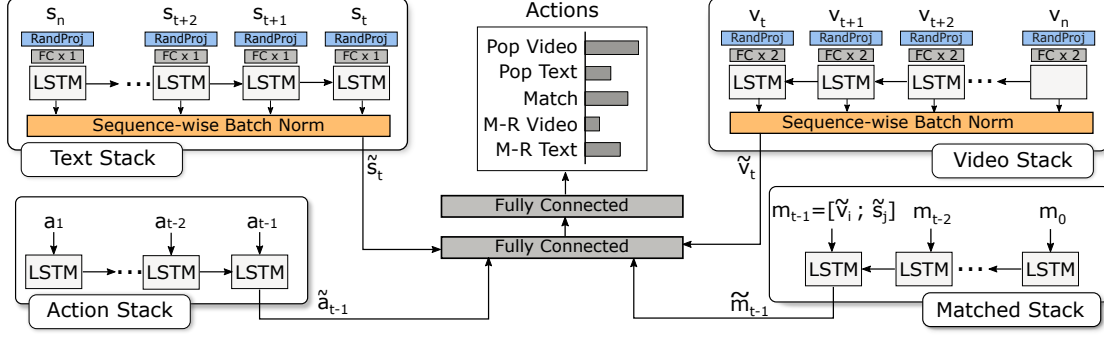
Figure 2: The Network Architecture. New components proposed in this paper are shown in blue and orange backgrounds.

Pop-Text action removes the topmost element from the Video Stack so that the second element can move up and get aligned. Similarly, the Pop-Video action removes the topmost element from the Text Stack. The Match action matches the two topmost elements, removes them from the Video and Text Stacks, and inserts them at the topmost position of the Matched Stack. These three actions are sufficient for one-to-one matchings. For the situation where one text element can match with more than one video element, the action Match-Retain-Text action removes the video element but leave the text element at the top of the Text Stack, so it may be matched with another video element. The matched elements are still inserted into the Matched Stack. The Match-Retain-Video action functions analogously.

3.2. Layer-wise Adaptive Rate Scaling (LARS)

Layer-wise Adaptive Rate Scaling (LARS) [64] scales the gradient updates for different layers so that the ratios between the update and the parameter magnitudes remain the same across layers. More formally, let $\boldsymbol{w}^l$ denote the parameters for the lth layer and $\nabla_{\boldsymbol{w}^l}\mathcal{L}$ denote the gradient of the loss function with respect to $\boldsymbol{w}^l$. The update $\Delta\boldsymbol{w}^l$ is calculated as

$$\Delta\boldsymbol{w}^l = \eta\|\boldsymbol{w}^l\|_2 \frac{\nabla_{\boldsymbol{w}^l}\mathcal{L}}{\|\nabla_{\boldsymbol{w}^l}\mathcal{L}\|_2} \tag{4}$$

where η is a global learning rate and $\|\cdot\|_2$ denotes the ℓ_2 norm. It is easy to see that, under LARS, $\Delta\boldsymbol{w}^l$ takes the direction of $\nabla_{\boldsymbol{w}^l}\mathcal{L}$ and has the magnitude of $\eta\|\boldsymbol{w}^l\|_2$. Thus, LARS ensures that the update is proportional to the parameters weights, so that layers with poorly scaled gradients can be optimized. In this paper, we apply LARS to gradient updates computed by Adam and replace the gradient term $\nabla_{\boldsymbol{w}^l}\mathcal{L}$ in Eq. 4 with the Adam update. LARS applies layer-wise scaling of the gradient whereas Adam applies component-wise scaling, obtaining complementary effects.

The effects of LARS have been mostly reported in large-batch training [64, 65]. In this paper, we utilize LARS to align the updates between network components dealing with different modalities.

3.3. Sequence-wise Batch Normalization

Similar to Batch Normalization (BN) [24], Sequence-wise Batch Normalization (SBN) [28] normalizes the input or the output of a recurrent neural network (RNN) across the batch and the temporal dimension. Let $\Phi \in \mathcal{R}^{B \times T \times P}$ be an order-3 tensor where B is the batch size, T the time steps, and P the dimension of the input vector at each time step. We write $\Phi_{b,t,p}$ for the scalar elements in the tensor. Assuming the number of unpadded elements in the tensor is M, we compute the mean μ_p, the variance σ_p^2, and the normalized tensor $\tilde{\Phi}$ as

$$\mu_p = \frac{1}{M} \sum_{b=1}^{B} \sum_{t=1}^{T} \Phi_{b,t,p} \tag{5}$$

$$\sigma_p^2 = \frac{1}{M} \sum_{b=1}^{B} \sum_{t=1}^{T} (\Phi_{b,t,p} - \mu_p)^2 \tag{6}$$

$$\text{SBN}(\Phi) = \tilde{\Phi}_{b,t,p} = \gamma_p \frac{\Phi_{b,t,p} - \mu_p}{\sqrt{\sigma_p + \epsilon}} + \beta_p \tag{7}$$

where ϵ is a small constant to prevent numeric instability. γ_p and β_p are per-dimension scaling and bias factors that restore network expressiveness after normalization.

To align the distributions of the video and the textual modalities, we apply sequence-wise batch normalization at the output of the respective LSTMs. Though the effects of normalization on convolutional and feedforward networks have been extensively studied (e.g., [23, 6, 45, 3, 8]), the recurrent variants (e.g., [12, 38, 67]) are less well understood.

3.4. Layer Normalization

Layer Normalization (LN) [4] is another common normalization technique for RNNs. Instead of normalizing across batch and time steps, it normalizes across the dimensions of the input vectors.

$$\mu_{b,t} = \frac{1}{P} \sum_{p=1}^{P} \Phi_{b,t,p} \qquad (8)$$

$$\sigma_{b,t}^2 = \frac{1}{P} \sum_{p=1}^{P} (\Phi_{b,t,p} - \mu_{b,t})^2 \qquad (9)$$

$$\text{LN}(\Phi) = \tilde{\Phi}_{b,t,p} = \gamma_p \frac{\Phi_{b,t,p} - \mu_{b,t}}{\sqrt{\sigma_{b,t} + \epsilon}} + \beta_p \qquad (10)$$

Preliminary experiments show that directly applying layer normalization to the LSTM outputs performed poorly. Instead, we employ the popular configuration from PyTorch FastRNN[2], which applies LN to the internal gates and the cell rather than the output.

$$\left[\boldsymbol{i}_t^\top, \boldsymbol{f}_t^\top, \tilde{\boldsymbol{c}}_t^\top, \boldsymbol{o}_t^\top \right]^\top = \text{LN}(W_x \boldsymbol{x}_t) + \text{LN}(W_h \boldsymbol{h}_{t-1}) \quad (11)$$

$$\boldsymbol{c}_t = \text{LN}(\sigma(\boldsymbol{f}_t) \otimes \boldsymbol{c}_{t-1} + \sigma(\boldsymbol{i}_t) \otimes \tanh(\tilde{\boldsymbol{c}}_t)) \qquad (12)$$

The hidden state computation (Eq. 3) remains unchanged. In the experiments, we find that LN provides good performance elevation, but not as effectively as SBN.

3.5. Random Projection

To reduce input dimensionality and trainable parameters, we adopt a simple random projection technique by Achlioptas [1]. Any input feature $\boldsymbol{x} \in \mathcal{R}^D$ is projected to P dimensions by multiplying with a random matrix $R \in \mathcal{R}^{P \times D}$, whose elements are random sampled as

$$R_{ij} = \begin{cases} \sqrt{3}, & \text{with probability } 1/6 \\ 0, & \text{with probability } 2/3 \\ -\sqrt{3}, & \text{with probability } 1/6 \end{cases} \qquad (13)$$

This projection has been shown to preserves distances between input vectors up to a constant factor $\sqrt{P}$ [1].

4. Experiments

In this section, we present the experimental results and analyze the effects of the three techniques of LARS, SBN, and random projection.[3]

4.1. Dataset and Performance Metrics

The YouTube Movie Summary dataset contains 93 short videos, each lasting about a few minutes for a total of 6.7 hours. The videos contain narration of the movie stories and selected clips from the original movie. The dataset was annotated at the sub-sentence level; a sentence can be broken

---

down into multiple snippets and aligned with different video segments. We follow the original split and use 66 videos for training, 12 for validation, and 15 for testing.

In preprocessing, we follow [15] and use a threshold-based method[4] for scene boundary detection to segment a video into a sequence of video clips. We set the threshold hyperparameter to 20 and the minimum length of a clip to 5 frames. We use the ground-truth chunking of sentences. Due to the existence of long sequences, we split training sequences so that each training sample contains 100 actions or less in order to improve training efficiency. The validation and testing sequences are unaffected.

Performance on this task is measured using video alignment accuracy and text alignment's overlap with ground truth. Since one video clip can match with at most one text snippet, video alignment accuracy is computed as the total length of correctly aligned video clips over the length of all clips. A text snippet can match with multiple video clips, so we compute the temporal intersection over union (IoU) of the predicted and the ground-truth clips matched to one snippet. Due to less-than-optimal video segmentation, the maximum training accuracies for video and text are 98.2% and 93.6% respectively.

4.2. Experimental Setup

We extract video and text features in the following manner. Following [2], we extract features from the central frame of each video clip using a Faster-RCNN [40] trained on the Visual Genome dataset [27]. For a text snippet, we extract 768-dimensional sentence embedding from the BERT-Base model [14]. After that, both features are reduced to 300 dimensions using either random projection (Section 3.5) or trainable dense layers. We normalized features to zero mean and unit variance before the trainable portion of the network.

Like the original NeuMATCH [15], we use 2 layers of LSTM in all stacks with 300 hidden dimensions for Video/Text Stacks, 20 for Matched Stack, and 8 for Action Stack. Additionally, we append ten positional features, such as the number of elements remaining in Video/Text Stacks, elements already matched, their ratios, and their reciprocals.

All experiments train for 350 epochs and use a batch size of 32. We use Adam as the optimizer and LARS is added to Adam for some networks. The norm of gradients is capped at 2. The learning rate is halved whenever training loss does not improve for 10 epochs. The best performing model, RP+LARS+SBN, adopts an initial global learning rate of 0.007 and label smoothing [49] of 3%. For comparison against pretraining on other multimodal data, we also use unimodal feature extractors from LXMERT [50] to generate features for the central frames of the video clips and

---

Figure 3: The training trajectory of the RP+LARS+SBN network. The x-axis indicates the number of epochs and the y-axis indicates accuracy / IoU.

the text snippets. See the supplemental material for more experimental details.

4.3. Main Results

In Table 1, we present the performance of different networks with and without pretraining on the LSMDC dataset. Naively omitting pretraining leads to severe overfitting; the validation set performance peaks at epoch 56, after which the performance drops. At epoch 350, there is a 3.6% performance gap in text IoU in comparison to [15]. The improvement in video accuracy is likely due to better input features. When we choose epoch 56 using validation performance (*i.e.* early stopping), the test performance gap can be reduced to 2.4%. When RP, LARS, and SBN are applied, we achieve the best performance of 30.6% video accuracy and 12.8% text IoU. The three techniques achieve strong regularization effects so that testing performance remains stable toward the end of the training (see Figure 3). With strong LXMERT features, we can obviate the need for pretraining on LSMDC, but this baseline is still inferior to the network using RP+LARS+SBN, suggesting RP+LARS+SBN may curtail overfitting in the sequence encoders. In the next few subsections, we present a detailed ablation study on the effects of the three techniques.

4.4. Effects of LARS

In order to study the regularization effects of the techniques, in this and all subsequent experiments, we train all models to maximum training accuracy under the current video segmentation (98.2% and 93.6% for video and text respectively). As a result, the test accuracy directly reflects the generalization gap at convergence. As the dataset is small, we find results from early stopping to have high variance and unreliable.

Table 2 shows the effects of LARS on test performance. Using the Adam optimizer alone leads to severe overfitting

Model	Video Acc.	Text IoU
NeuMATCH w/ pretraining [15]	12.0	10.4
Our pretraining	26.8	10.3
No pretraining	24.2	6.8
No pretraining + early stopping	24.9	8.0
No pretraining + LXMERT features	28.5	10.8
No pretraining + RP+LARS+SBN	**30.6**	**12.8**

Table 1: Test performance when NeuMATCH is trained with or without pretraining on LSMDC and the regularization techniques RP, LARS, and SBN.

Optimizer	Video Acc.	Text IoU
Adam	21.7	8.0
Adam + warm-up	24.7	9.3
LARS	**30.6**	**12.8**

Table 2: Comparing LARS to Adam and learning rate warm-up on test performance. All model use randomly projected features and SBN.

(21.7% video accuracy and 8.0% text IoU). LARS reduces the generalization gap by 8.9% for video accuracy and 4.8% for text IoU.

We further inspect the ratio between the norm of model parameters $\|\boldsymbol{w}\|_2$ and the gradient $\|\nabla_{\boldsymbol{w}}\mathcal{L}\|_2$ in the Video Stack, the Text Stack, the Action/Matched Stacks, and the two FC layers before the final softmax. A higher ratio indicates the network is closer to a local minimum. Results are shown in Figure 4 and Table 3.

Comparing the three models in Table 2, we observe that LARS provides substantial benefits in the optimization of the network; LARS always achieves higher ratios and hence better convergence than Adam + warm-up and Adam. In particular, with both RP and SBN, applying LARS causes a 20-fold increase of the ratio over Adam+warm-up for the Action / Matched Stack. This indicates these two stacks converge significantly better with LARS than without. We further note that the gradient norm ratios under LARS increase sharply after 200 epochs. Observing the training trajectory of RP+LARS+SBN in Figure 3, the network is getting very close to 100% training accuracy at this time, yet further optimization is still beneficial. This indicates that LARS provides good optimization even when the network approaches the minimum and the gradient becomes small.

It may seem counter-intuitive that better optimization can narrow the generalization gap. We contend that LARS contributes to generalization because it aligns the magnitudes of the gradient updates and allows the four stacks
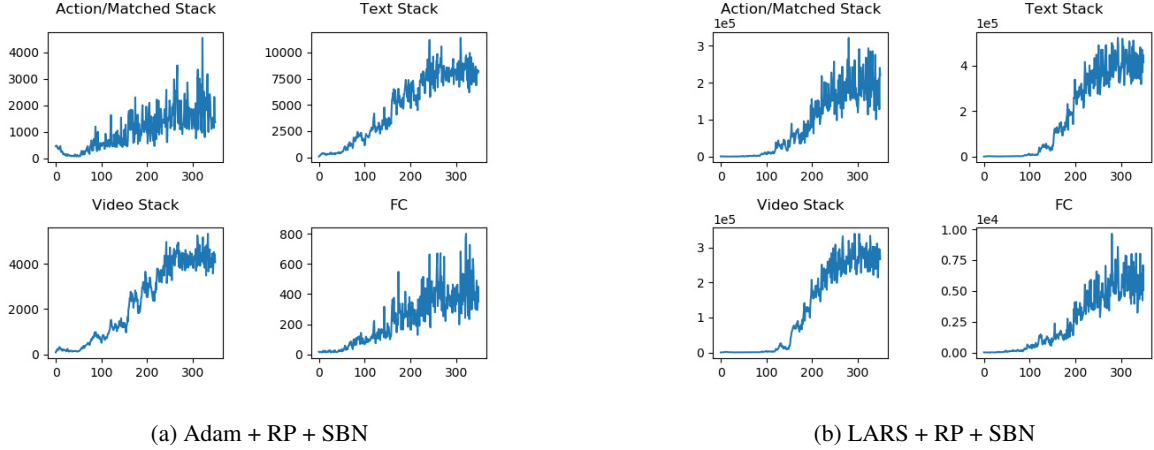
(a) Adam + RP + SBN

(b) LARS + RP + SBN

Figure 4: The ratio $\frac{\|\boldsymbol{w}\|_2}{\|\nabla_w \mathcal{L}\|_2}$ between the parameter norm and the gradient norm in four network components. A greater ratio indicates the model is closer to a local minimum.

Model	Video Stack	Text Stack	Text Stack / Video Stack	Action & Matched Stacks	FC
RP + Adam + SBN	4263.5	8075.0	1.9	1711.9	432.1
RP + Adam + warm-up + SBN	10378.1	23305.1	2.2	8228.3	2120.8
RP + LARS + SBN	2.7×10^5	4.1×10^5	1.5	1.9×10^5	5604.5
RP + Adam + 2×LN	2.6×10^6	5.5×10^6	2.1	1.1×10^6	7.4×10^4
RP + LARS + 2×LN	3.5×10^6	6.4×10^6	1.8	1.4×10^6	9.0×10^4
RP + Adam + 4×LN	2.3×10^6	3.8×10^6	1.7	7.3×10^5	7.8×10^4
RP + LARS + 4×LN	1.3×10^7	2.1×10^7	1.6	1.0×10^7	2.5×10^5

Table 3: Means of the gradient norm ratio $\frac{\|\boldsymbol{w}\|_2}{\|\nabla_w \mathcal{L}\|_2}$ in the last 20 training epochs for different models and model components.

to train at similar speed. As a result, the FC layers do not overfit to any particular stack. This argument is supported by two facts. First, the Action/Matched Stacks are much better optimized under LARS. Second, compared to Adam, LARS always reduces the relative gap between the $\|\boldsymbol{w}\|_2/\|\nabla_{\boldsymbol{w}}\mathcal{L}\|_2$ ratios of the Video Stack and the Text Stack. For example, with RP+Adam+SBN, the ratio of the Text Stack is 1.9 times the ratio of the Video Stack. With RP+LARS+SBN, this reduces to 1.5. This suggests the optimization of the two modalities is better aligned with LARS than without.

4.5. Effects of Normalization

We compare SBN against the popular layer-normalized LSTMs. In the 2×LN version, we apply LN to the Video and Text Stacks. In the 4×LN version, all four stacks are layer-normalized.

Table 4 shows that when LARS is applied, SBN outperforms 4×LN, which slightly outperforms 2×LN. Observing Table 3, we notice that LARS+LN can push the gradient norm ratios even higher than LARS+SBN. SBN creates

noise in the gradient as data points in the same batch are randomly sampled. The noise may have prevented extremely close convergence or falling into sharp minima [47], thereby providing some regularization effects.

With Figure 5, we additionally examine the mean and variance of the features provided by the Video and Text Stacks when we apply SBN, 2×LN, and no normalization. Under SBN, the means are very close to zero but the variances are much higher than the other versions. This suggests the features are more informative under SBN because they change more for different input. Even though the SBN contains a bias factor that may shift the mean away from zero, the network does not learn to do so.

4.6. Effects of Random Projection

Table 5 compares models using features in the original dimensions (Full) and randomly projected (RP) features. To create fair comparisons in this experiment, the only difference within each Full-versus-RP pair is whether the FC layers before the Stacks are trainable or fixed random projection matrices. The models use different normalization tech-
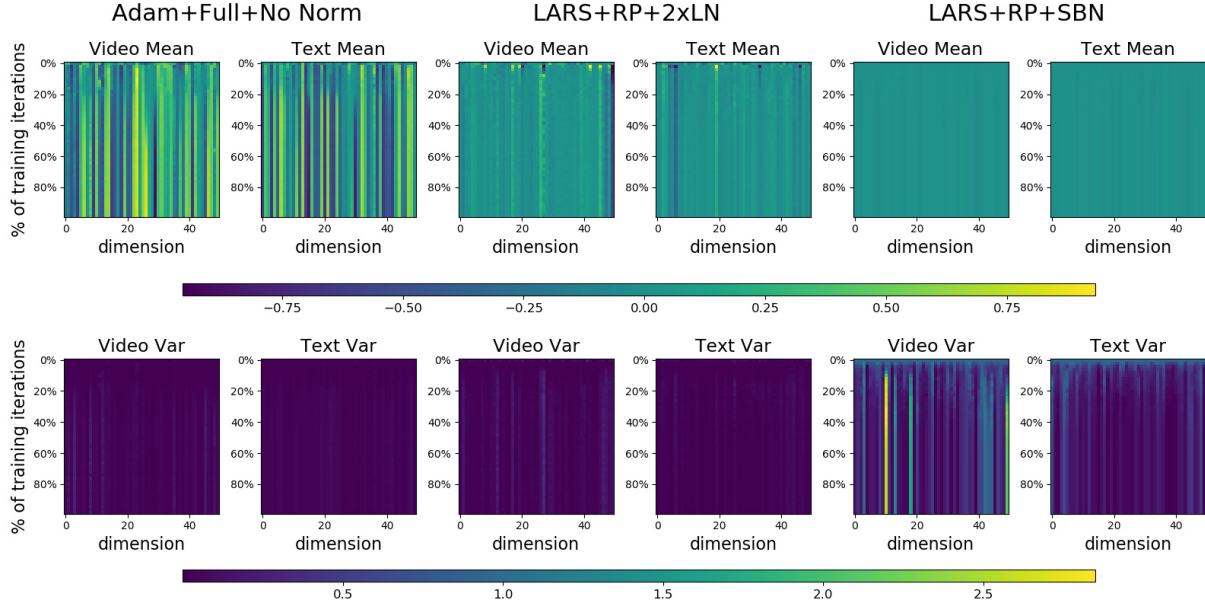
Figure 5: The mean and variance distributions of the internal video and text feature, which are the outputs of the Video and Text Stacks. We plot only the first 50 dimensions throughout the training process and compare SBN against the network without any normalization.

Normalization	Optimizer	Video Acc.	Text IoU
SBN	Adam	21.7	8.0
2×LN	Adam	23.3	8.2
4×LN	Adam	23.6	7.5
SBN	LARS	**30.6**	**12.8**
2×LN	LARS	25.3	8.8
4×LN	LARS	25.9	8.7

Table 4: Comparison of different normalization techniques. Random projection is used for all models in this table.

Feature / Norm	Video Acc.	Text IoU	# Parameters
Full + SBN	27.6	9.9	4.25M
RP + SBN	**30.6**	**12.8**	3.41M
Full + 2×LN	25.4	8.3	4.26M
RP + 2×LN	25.3	8.8	3.42M
Full + 4×LN	26.9	9.9	4.26M
RP + 4×LN	25.9	8.7	3.42M

Table 5: Comparison between full and RP features and the number of trainable parameters. All models use LARS.

niques, SBN, 2×LN, and 4×LN and are always optimized with LARS. RP features usually obtain better performance over the model using full features, though full features per-

form better for 4×LN. The regularization efforts of RP can be attributed to the fact that RP reduces the number of trainable parameters by 19% in the models we tested.

5. Conclusions

Establishing fine-grained correspondences between the vast reservoir of parallel movie videos and screenplays can empower many practical applications. A major roadblock is the limited availability of annotated data. The large LSMDC dataset deviates from realistic situations; the YMS dataset is more realistic but 20 times smaller. With this paper, we aim to improve data efficiency of sequence alignment networks, which helps in removing the need to pretrain on the LSMDC dataset.

The three techniques, namely layer-wise adaptive rate scaling, sequence-wise batch normalization, and random projection of input features, allow the network to beat the previous best result, which was pretrained on multimodal data. Experiments suggest that LARS helps in aligning the pace of learning in different network components. We believe this work represents a step toward data-efficient multimodal learning.

References

[1] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, 2001.

[2] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and VQA. *arXiv preprint arXiv:1707.07998*, 2017.

[3] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. 2019.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. 2016.

[5] Yogesh Balaji, Martin Renqiang Min, Bing Bai, Rama Chellappa, and Hans Peter Graf. Conditional gan with discriminative filter generation for text-to-video synthesis. In *IJCAI*, pages 1995–2001, 2019.

[6] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *NeurIPS*. 2018.

[7] Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. Weakly-supervised alignment of video with text. In *ICCV*, pages 4462–4470, 2015.

[8] Yongqiang Cai, Qianxiao Li, and Zuowei Shen. A quantitative analysis of the effect of batch normalization on gradient descent. *arXiv 1810.00122*, 2018.

[9] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3TW: Discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *CVPR*, 2019.

[10] Jingyuan Chen, Xinpeng Chen, Lin Ma, Zequn Jie, and Tat-Seng Chua. Temporally grounding natural sentence in video. In *EMNLP*, 2018.

[11] Zhenfang Chen, Lin Ma, Wenhan Luo, and Kwan-Yee K. Wong. Weakly-supervised spatio-temporally grounding natural sentence in video. In *ACL*, 2019.

[12] Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. In *ICLR*, 2016.

[13] Timothee Cour, Chris Jordan, Eleni Miltsakaki, and Ben Taskar. Movie/script: Alignment and parsing of video and text transcription. *ECCV*, pages 158–171, 2008.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Pelin Dogan, Boyang Li, Leonid Sigal, and Markus Gross. A neural multi-sequence alignment technique (NeuMATCH). In *CVPR*, 2018.

[16] Pelin Dogan, Leonid Sigal, and Markus Gross. Neural sequential phrase grounding (SeqGROUND). In *CVPR*, 2019.

[17] Mark Everingham, Josef Sivic, and Andrew Zisserman. Hello! My name is... Buffy—automatic naming of characters in TV video. In *BMVC*, 2006.

[18] Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. TALL: Temporal activity localization via language query. In *ICCV*, 2017.

[19] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, 2017.

[20] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. In *ICANN*, 1999.

[21] Dongliang He, Xiang Zhao, Jizhou Huang, Fu Li, Xiao Liu, and Shilei Wen. Read, watch, and move: Reinforcement learning for temporally grounding natural language descriptions in videos. In *AAAI*, 2019.

[22] Lisa Anne Hendricks, Oliver Wang, Eli Shechtman, Josef Sivic, Trevor Darrell, and Bryan Russell. Localizing moments in video with temporal language. In *EMNLP*, 2018.

[23] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. *arXiv 1803.01814*, 2018.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network trainingby reducing internal covariate shift. In *ICML*, 2015.

[25] J. Ji, K. Cao, and J. C. Niebles. Learning temporal action proposals with fewer labels. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7072–7081, Oct 2019.

[26] Nam Wook Kim, Benjamin Bach, Hyejin Im, Sasha Schriber, Markus Gross, and Hanspeter Pfister. Visualizing nonlinear narratives with story curves. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):595–604, 2018.

[27] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 2017.

[28] César Laurent, Gabriel Pereyra, Philémon Brakel, Ying Zhang, and Yoshua Bengio. Batch normalized recurrent neural networks. *arXiv 1510.01378*, 2015.

[29] Alistair Letcher, David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. Differentiable game mechanics. *Journal of Machine Learning Research*, 20:1–40, 2019.

[30] Yitong Li, Martin Renqiang Min, Dinghan Shen, David Carlson, and Lawrence Carin. Video generation from text. *arXiv preprint arXiv:1710.00421*, 2017.

[31] Tianwei Lin, Xu Zhao, Haisheng Su, Chongjing Wang, and Ming Yang. Bsn: Boundary sensitive network for temporal action proposal generation. In *ECCV*, 2018.

[32] Yuan Liu, Lin Ma, Yifeng Zhang, Wei Liu, and Shih-Fu Chang. Multi-granularity generator for temporal action proposal. In *CVPR*, 2019.

[33] Yue Liu, Xin Wang, Yitian Yuan, and Wenwu Zhu. Cross-modal dual learning for sentence-to-video generation. In *ACM MM*, 2019.

[34] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Jason Li, Taroon Bharti, and Ming

Zhou. Univl: A unified video and language pre-training model for multimodal understanding and generation. *arXiv 2002.06353*, 2020.

[35] Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. End-to-end learning of visual representations from uncurated instructional videos. In *CVPR*, 2019.

[36] Iftekhar Naim, Young C. Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *NAACL*, 2015.

[37] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. Sceneskim: Searching and browsing movies using synchronized captions, scripts and plot summaries. In *UIST*, page 181–190, New York, NY, USA, 2015.

[38] Mengye Ren, Renjie Liao, Raquel Urtasun, Fabian H. Sinz, and Richard S. Zemel. Normalizing the normalizers: Comparing and extending network normalization schemes. In *ICLR*, 2017.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[41] Philipp Rimle, Pelin Dogan, and Markus Gross. Enriching video captions with contextual text. *arXiv 2007.14682*, 2020.

[42] Anna Rohrbach, Atousa Torabi, Marcus Rohrbach, Niket Tandon, Christopher Pal, Hugo Larochelle, Aaron Courville, and Bernt Schiele. Movie description. *IJCV*, 123(1):94–120, May 2017.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015.

[44] Pramod Sankar, CV Jawahar, and Andrew Zisserman. Subtitle-free movie to script alignment. In *BMVC*, 2009.

[45] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *NeurIPS*, pages 2483–2493. 2018.

[46] Haoyue Shi, Jiayuan Mao, Kevin Gimpel, and Karen Livescu. Visually grounded neural syntax acquisition. In *ACL*, 2019.

[47] Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *ICLR*, 2018.

[48] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV*, 2019.

[49] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016.

[50] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019.

[51] Niket Tandon, Gerard de Melo, Abir De, and Gerhard Weikum. Knowlywood: Mining activity knowledge from hollywood narratives. In *CIKM*, 2015.

[52] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. Story-based video retrieval in tv series using plot synopses. In *International Conference on Multimedia Retrieval*, 2014.

[53] Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. Book2movie: Aligning video scenes with book chapters. In *CVPR*, pages 1827–1835, 2015.

[54] Bairui Wang, Lin Ma, Wei Zhang, and Wei Liu. Reconstruction network for video captioning. In *CVPR*, 2018.

[55] Josiah Wang and Lucia Specia. Phrase localization without paired training examples. In *ICCV*, 2019.

[56] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multi-modal classification networks hard? In *CVPR*, 2019.

[57] X. Wang, W. Chen, J. Wu, Y. Wang, and W. Y. Wang. Video captioning via hierarchical reinforcement learning. In *CVPR*, 2018.

[58] Bo Wu, Haoyu Qin, Alireza Zareian, Carl Vondrick, and Shih-Fu Chang. Analogical reasoning for visually grounded language acquisition. *arXiv 2007.11668*, 2020.

[59] Hao Wu, Jiayuan Mao, Yufeng Zhang, Yuning Jiang, Lei Li, Weiwei Sun, and Wei-Ying Ma. Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations. In *CVPR*, 2019.

[60] Jie Wu, Guanbin Li, Si Liu, and Liang Lin. Tree-structured policy based progressive reinforcement learning for temporally language grounding in video. In *AAAI*, 2020.

[61] Huijuan Xu, Kun He, Bryan A. Plummer, Leonid Sigal, Stan Sclaroff, and Kate Saenko. Multilevel language and vision integration for text-to-clip retrieval. In *AAAI*, 2018.

[62] Huijuan Xu, Boyang Li, Vasili Ramanishka, Leonid Sigal, and Kate Saenko. Joint event detection and description in continuous video streams. In *WACV*, 2019.

[63] Guangnan Ye, Yitong Li, Hongliang Xu, Dong Liu, and Shih-Fu Chang. Eventnet: A large scale structured concept library for complex event detection in video. In *ACM MM*, 2015.

[64] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv 1708.03888*, 2017.

[65] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. *arXiv 1904.00962*, 2019.

[66] Haonan Yu and Jeffrey Mark Siskind. Sentence directed video object codiscovery. *International Journal of Computer Vision*, 124(3):312–334, 2017.

[67] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *NeurIPS*. 2019.

[68] Da Zhang, Xiyang Dai, Xin Wang, Yuan-Fang Wang, and Larry S. Davis. MAN: Moment alignment network for natural language moment retrieval via iterative graph adjustment. In *CVPR*, 2019.

[69] Luowei Zhou, Yingbo Zhou, Jason J. Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *CVPR*, 2018.

[70] Linchao Zhu and Yi Yang. Actbert: Learning global-local video-text representations. In *CVPR*, 2020.

[71] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, pages 19–27, 2015.