期中项目
CNN 模型与 RNN 模型实现

冯家苇 *　　黄杨峻*

数据科学与计算机学院　计算机科学与技术

17341035，　　17341059

1 实验介绍

本实验包含两个任务, 一是用卷积神经网络 (CNN) 对 CIFAR-10 数据集进行标签预测, 一是用循环神经网络 (RNN) 对 STS Benchmark 数据集进行文本相似度的预测.

2 算法原理

2.1 卷积神经网络

卷积神经网络 (CNN) 跟普通神经网络非常相似, 都由具有可学习的权重和偏置常量神经元组成, 是一种十分经典的前馈神经网络. 基于较为简单的 Lenet 如图1，我们做以下说明：

CNN 通常包含以下几种结构:

- 卷积层 (Convolutional layer)
  卷积神经网络种每层卷积层由若干卷积单元组成, 卷积运算的目的是提取输入的不同特征, 由卷积操作, 从低级的特征 (如边缘, 线条, 角点) 到高级的特征 (特定弧线, 形状) 都能被提取出来, 卷积操作的公式为

$$y_{i,j} = \sum_{u=1}^{m}\sum_{v=1}^{n} w_{u,v} \times x_{i-u+1,j-v+1}$$

  其中, Y 为卷积层的输出,W 为大小 $m \times n$ 的卷积核, X 为原图像.
  卷积层种体现了 CNN 在结构上的两种特性

  1. 局部连接 (Local Connectivity)
     普通神经网络直接用全连接的方法吧隐层和输出层连接, 如此就会产生大量的边, 使到反向传播的过程特别慢. 卷积层则令每个隐含单元只能连接输入单元的一部分, 这个输入区域大小也叫神经元的感受野 (receptive field). 这个感受野可能是输入图像的一小片邻域, 也可能时一段音频在某个时间段上的信号.
  2. 权重共享 (Parameter Sharing)
     应用参数共享, 可以大幅减少参数数量. 我们把同一深度的平面叫做深度切片 (depth slice), 那么同一个切片应该共享同一组权值和偏置. 为什么我们可以这样做呢? 一方面同一图层种, 某一点上的特征很重要, 这个特征在另一个点上也同样重要; 另一方面, 减少模型的规模也能避免过拟合, 让模型具有更高的泛化能力.

*按学号顺序排序，不代表贡献大小

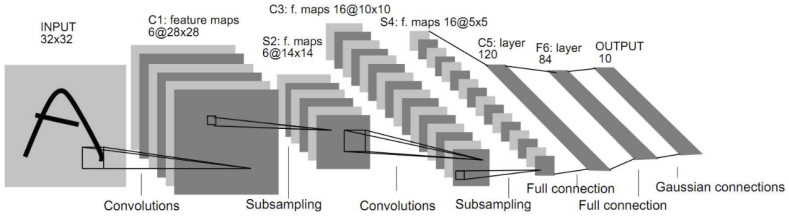


图 1: Lenet Framework

- 线性整流层 (ReLU layer)
  ReLU 层用于加入非线性因素, 通常来说用 ReLU 层也可以用 Tanh 层代替, 如果没有非线性激励函数, 那无论有多少层神经网络输出都是输入的线性组合, 线性模型的表达能力是远远不够的.
  ReLU 函数十分简单, 当输入大于零时输入等于输出, 否则输出等于 0.
- 池化层 (Pooling layer)
  这里就涉及到了卷积神经网络的第三个结构特征: 空间上的下采样 (Down Sampling)
  下采样的目的很简单, 就是为了减少特征, 进而提取出更为重要的特征.
  通常, 池化层有两种

  1. Max Pooling
     取四个点的最大值. 这是最常见的池化策略.2014 年提出的 Fractional Max Pooling 模型一度霸占了 cifar10 benchmark 的首位.
  2. Mean Pooling
     取四个点的均值. 也挺常见.

- 全连接层
  与普通的神经网络全连接层定义类似, 不在此赘述.

2.2 循环神经网络

循环神经网络 (RNN) 是一类扩展的人工神经网络, 它是为了对序列数据进行建模而产生的. 它的核心思想是通过神经网络在时序上的展开, 找到样本之间的序列相关性. 它比起 DNN,CNN 等网络的不同之处就在于此: 不仅仅考虑前一时刻输入, 而且赋予了网络对之前的内容' 记忆' 功能. 通常来说,RNN 的应用场景很广泛, 主要是输入为时序序列的场景, 例如: 视频处理, 股价预测, 机器翻译, 语音识别, 图像描述生成, 文本相似度计算等等.

RNN 主要由输入层, 隐层和输出层组成, 当前时间节点的隐层与下一个时间节点的隐层有边相连, 这条边主要用于实现时间记忆功能. 我们可以用一组方程来表示 RNN 的迭代和输出:

$$\begin{aligned} h_t &= Ux_t + Wh_{t-1}, \\ s_t &= f(h_t) \\ o_t &= g(Vs_t) \end{aligned} \tag{1}$$

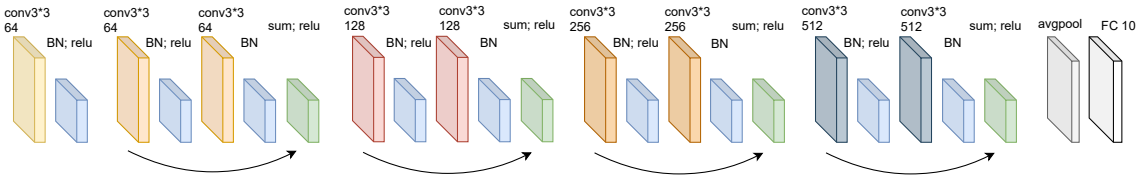


图 2: Resnet-18

其中 f 和 g 均为激活函数,f 可以是 tanh,relu,sigmoid 等激活函数,g 通常是 softmax 也可以为其他函数. 在这里,W,U,V 每个时刻都是权重共享的. 隐层状态 S 可以理解为 f(t 时刻输入+(1,2,..t-1) 时刻的状态总结)

值得一提的是 RNN 的反向传播, 我们注意到每次输出值 o_t 都会产生误差 e_t, 总的误差可以表示为 $E = \sum_t e_t$, 进而我们可以用 cross entropy 或者 MSE 来计算损失函数. 由于每一步输出还取决于之前若干步的网络状态, 我们需要对 BP 进行改造, 这个改造过的算法叫做 BPTT, 就是将输出端的误差反向传递, 运用梯度下降法进行更新.

RNN 有个严重的问题, 就是梯度小时或者梯度爆炸的问题 (BPTT 特性和长时间序列所导致), 时间过长, 记忆值难免过小, 因此我们有 LSTM 和 GRU 算法对梯度小时和梯度爆炸问题的改进.

2.3 残差神经网络

1. 深度网络的退化问题

   卷积神经网络以其独有的网络深度与宽度学习到了更大感受野的更高层的语义知识。但是过深的网络往往会导致网络退化问题（Degenerate Problem）：网络深度增加，但是无论训练还是测试的准确度均下降。导致这个问题的原因可能是梯度方向随着网络深度增加变得不准。

2. 残差学习

   残差学习基于这样一个先验假设：对于一个浅层网络，若想通过增加其层数建立深层网络，一个极端情况是增加的层全为恒等映射（Identity Mapping），即深层网络至少可以与浅层网络的性能一样好。

   基于此，残差学习可认为是对于一个模块，当输入为 x 时输出为 H(x)，现在我们希望学习其残差 F(x)=H(x)-x，这样输出便由 x 与 F(x) 加和而成了。

3. 残差神经网络（Deep Residual Network）

   Resnet 基于以下的设计原则：当特征图（feature map）大小降低一半时，特征图数量增加一倍。例如 Resnet18 的模型见图2。

2.4 长短时记忆网络（**LSTM**）

与基本的 RNN 模型不同的是，LSTM 具备学习长期信息的能力。LSTM 的模型结构如图3所示。

1. 隐藏状态

   隐藏状态由模型下方的直线形式化表示。与之相连的四个激活函数的输入为上一隐藏状

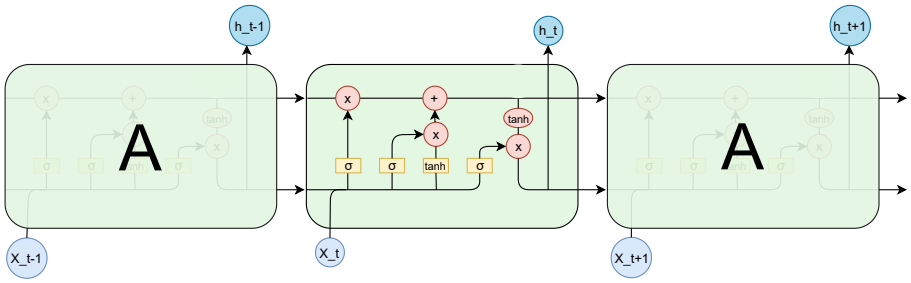


图 3: LSTM

态与当前输入拼接而成。其计算公式如下：

$$\begin{aligned} f_t &= \sigma(W_f * concate(h_{t-1}, x_t) + b_f) \\ i_t &= \sigma(W_i * concate(h_{t-1}, x_t) + b_i) \\ \hat{C}_t &= \tanh(W_C * concate(h_{t-1}, x_t) + b_C) \\ o_t &= \sigma(W_o * concate(h_{t-1}, x_t) + b_o) \end{aligned} \tag{2}$$

2. 细胞状态

细胞状态可由一条贯穿整个结构的直线形式化表示。第一个乘号相当于一个与门，将上一个细胞状态和一个经过 sigmoid 函数激活后的值相与，代表历史信息的选择性通过，称其为“遗忘门”。

第二个加号相当于一个或门，即简单相加。它的另一个输入也是通过选择性记忆的方式生成。

当前细胞状态除了要传递给下一状态，同时需要通过选择性记忆的方式生成当前状态的隐藏状态，对最后一层而言即输出状态。

以上可用下面的公式形式化表述：

$$\begin{aligned} c_t &= f_t * C_{t-1} + i_t * \hat{C}_t \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{3}$$

2.5 孪生网络（Siamese Network）

Siamese 网络是一种相似度的度量方法。它的主要思想是将样本通过特征变换映射到目标空间，在目标空间使用简单的距离度量方法计算样本间相似度。这种网络结构淡化了分类标签的效用，使其具有较好的可扩展性。

3 方法细节

3.1 CNN 部分

3.1.1 算法流程

首先, 用 torchvision.dataset 里的 CIFAR10 下载数据, 然后对数据集采用如下变换:

1. 以 size=32,padding=4 的参数随机裁剪图片.(数据增强)
2. 以 p=0.5 的概率随即翻转.(数据增强)
3. 归一化数据 mean=(0.4914, 0.4822, 0.4465), std=(0.2023, 0.1994, 0.2010)

然后, 我们用 batch_size=128 的大小载入 dataloader. 紧接着, 我们分别定义了 LeNet-5 和 ResNet18 进行训练, 具体超参为学习率为 0.01, 使用的优化器为随机梯度下降优化器 (SGD), 损失函数为交叉熵, 同时引入动量法 (衰减率为 0.9) 进行训练, 进行的 epoch=200.

3.1.2 小 trick

1. 用 CELU 代替 RELU, 这样的激活函数更加光滑, 比起在原点处不连续的 RELU 更有助于泛化. 下图为 CELU 函数和 RELU 函数的对比, 可以清楚看到 CELU 在原点是不连续的.

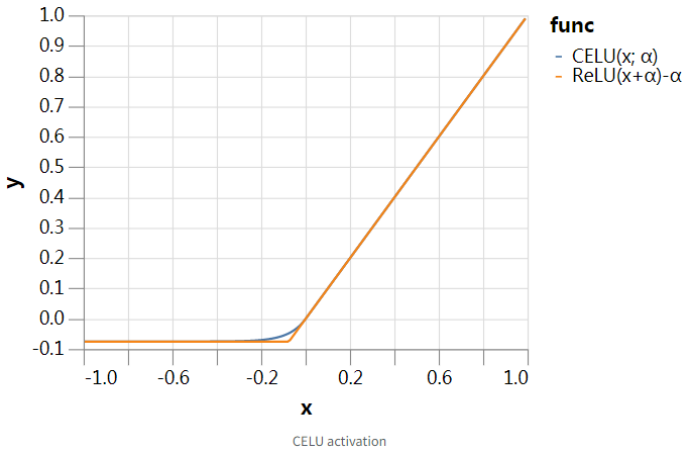


图 4: celu

2. 使用 label smoothing 技术 [1], 该技术的本质就是将标签变得平滑, 让预测结果不那么相信所属的类别, 起到增强鲁棒性的作用. 原来交叉熵损失函数可以表示为

$$loss = -\sum_{k=1}^{K} q(k|x)log(p(k|x))$$

其中,K 为样本大小. 为了使得模型 less confident, 我们提出以下机制

$$q'(k|x) = (1-\epsilon)\delta_{k,y} + \epsilon u(k)$$

把 q(k) 函数改为 q'(k), 代入交叉熵公式来替代原来的 loss function.

3.2 RNN 部分

3.2.1 Siamese-biLSTM-NP

针对文本相似度分析的任务,基于 [2] 使用的度量方法,这里使用了如图5的 Siamese-biLSTM-NP 模型。

首先输入两个文本的词向量表示 (batch, textlen, inputchannel)，对这两个文本使用同一个网络（即参数共享）计算：首先过一个 biLSTM 得到隐藏层的输出 (batch, textlen, outputchan-

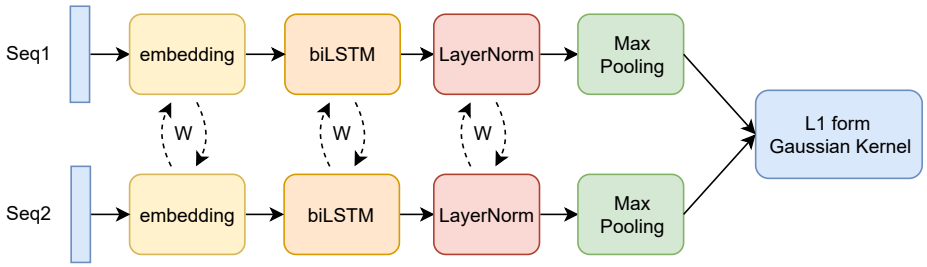


图 5: Siamese-biLSTM-NP Framework

nel)，然后对其做 LayerNorm，然后对 textlen 做一个 max-pooling 得到文本的向量表示 (batch, outputchannel)。最后计算两个样本通过一范数形式的高斯核结果作为相似度度量。

- end-to-end model
  在 pytorch 框架 [3] 下，我们将词向量设置为可学习的，使得网络可以完成一个端到端的训练。
- biLSTM
  考虑到序列后部可能对序列前部产生影响，双向的长短时记忆网络可以更好的对上文与下文的信息进行建模，间接提高了模型的感受野。但是，注意到本次任务主要由短句组成，在实验中可以发现使用双向网络并不能带来特别显著的提升。
- LayerNorm
  归一化是一种很好的抑制梯度消失的方法，它提供了一种良好的线性性质与非线性性质中和的机制。批归一化（Batch Normalization）对处理序列化数据的网络不大适用，原因是不同序列的长度不同，对做 batch 间的归一化不能反映统计信息的全局分布。
  在使用层归一化的序列模型中，我们在每个时间片内做归一化，统计隐层节点的均值与方差，使得归一化计算与 batch size 无关了。
- max over-time pooling
  作为一个常见的 trick，这里在时间片，即 textlen 维度进行 max pooling 操作。
- gaussian kernel in L1 norm form
  对于长度不相等的两个样本 a，b，L1 范数下的高斯核函数公式如下：

$$g(h^a_{T_a}, h^b_{T_b}) = \exp(-||h^a_{T_a} - h^b_{T_b}||_1) \tag{4}$$

  这种简单的处理方式使得 LSTM 更为直接而全面的捕捉样本间的语义差异点，Lecun 等人在 [4] 中指出了 L2 范数会使得当取平方时，参数的梯度会随着输入趋近于 0 而消失，导致损失函数难以优化。

3.2.2 数据预处理

使用 gensim 库的 Word2Vec 模型在训练集上做词向量的预训练。将预训练结果作为嵌入层的初始化参数。

3.2.3 学习率调整机制

我们使用了 pytorch 的 lr_scheduler 方法，设置忍受值 patience=20，当 20 轮 epoch 内指定的 validation loss 没有下降，则将学习率除以 10。

3.2.4 其他 trick

- Label 调整
  由于 STS 任务中的标签是一个 0 到 5 的连续值，实验中发现，相比在模型最后的输出结果 $y \in [0, 1]$ 乘以 5，将 label 做一个尺度变换为 0 到 1，即除以 5 具有更好的效果，原因可能是使得梯度变得更平滑。
- padding 方法与 embedding 矩阵的调整
  我们尝试过处理视频数据常用的 Temporal Random Crop 方法，随机选取连续定长的样本数据，并采用回滚的方式补齐为同等长度，但是实验中结果并不好，因此我们使用最简单的补“0”的方法，在 embedding 矩阵中设置一个空的词向量，补齐用的索引指向这个向量。

4 实验结果

4.1 CNN 部分

4.1.1 实验结果对比

经过训练, 我们最后得到了结果,LeNet 在训练集的准确率和测试集的准确率都比不上 ResNet18, 而且所需要大约 epoch=80 的训练时间才能拟合到最佳状态, 此后便会过拟合, 而 ResNet18 的性能优越, 在 epoch=30 时就能达到收敛状态. 不过, 由于模型复杂度的原因,ResNet 的训练时间远大于 LeNet.

表 1: LeNet:lr = 0.01, weight_decay=5e-4, epoch=80

Class	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
Accuracy	73%	93%	88%	47%	71%	50%	67%	92%	91%	95%
Train accuracy	74.2%									
Valid accuracy	71.7%									

表 2: ResNet:lr = 0.01, weight_decay=5e-4, epoch=30

Class	plane	car	bird	cat	deer	dog	frog	horse	ship	truck
Accuracy	99%	97%	95%	88%	90%	82%	84%	98%	96%	98%
Train accuracy	95.6%									
Valid accuracy	88.9%									

图6、7分别为用 tensorboard 绘制的两模型的 Loss 收敛曲线:

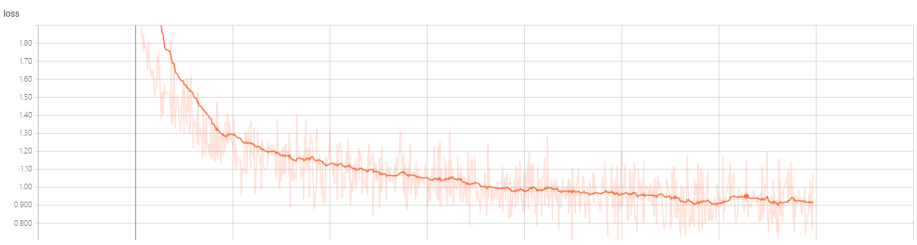


图 6: LeNet-loss

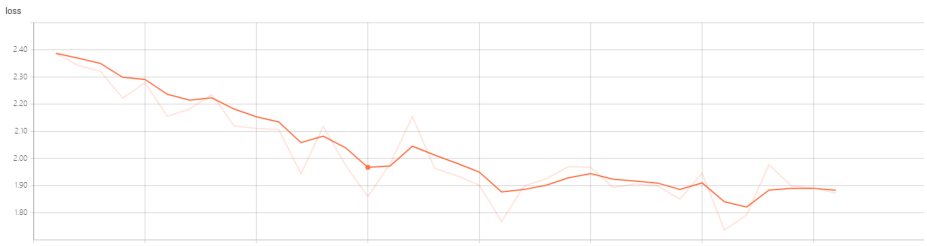


图 7: ResNet-loss

我们将最终得到的第一层卷积层权重做可视化，结果如下图8所示。

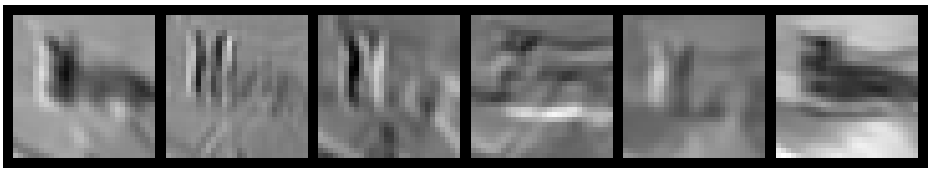


图 8: Conv2d-weight

4.1.2 Ablation Study

这次实验中, 我们还对 ResNet 模型加入了两个 trick, 分别是 CeLU 函数代替 ReLU 函数和 LS(Label Smoothing). 实验效果如下:

表 3: Ablation Study

Strategy	Res18	Res18+CeLU	Res18+LS	Res18+CeLU+LS
Train Accuracy	95.6%	95.7%	95.4%	95.2%
Valid Accuracy	88.9%	88.5%	88.8%	88.7%
Convergent Epoch	30	25	27	23

可以看出,trick 的引入几乎对模型准确率影响不是太大, 但训练时收敛所需的 epoch 明显减少了, 这对训练的效率提升很有帮助.

4.2 RNN 部分

4.2.1 数据集：STSbenchmark

STSbenchmark 是一个英语数据集，全称 Semantic Textual Similarity 2012-2017 Dataset。它含有三个 Topic 共计 8628 个样本，划分为 train，development，test 三个数据集，大小分别为 5749，1500，1379，每个样本由两个短句组成，并分有一个 0 到 5 的标签，表示两个短句的相似程度。

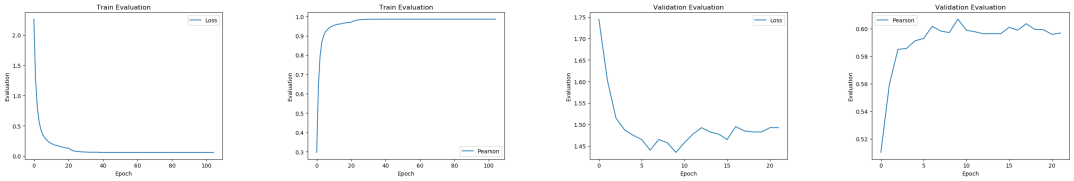
4.2.2 实验设置

按照 gensim 官方文档的建议，我们将 embedding 词向量的维数设置为 100，最小出现次数为 3。padding 后的样本长度统一设置为 55。由于数据集样本的长度较短，同时为了减缓过拟合，我们使用了单层的双向 LSTM，隐藏节点设置为 64 个；这里选用的优化器是 REMprop，学习率为 0.001。损失函数为 MSE 均方误差。

4.2.3 实验评估

参考 [5]，这里使用的评价指标为 Pearson 相关系数。我们的模型在 dev 数据集上达到了 60.7%，test 上达到了 57.6%。由于我们对 Label 做了尺度缩放，这里我们在展示 Loss 时还原了尺度。实验结果如下图9。

4.2.4 实验评估



(a) (b) (c) (d)

图 9: 评估结果

4.2.5 Ablation Study

这里将我们的模型与简化模型做了对比实验，部分实验结果如下表（具体见代码文件）：

表 4: Ablation Study

model	Test Loss	Test Pearson
LSTM(1 layer, fixed embedding)	3.05	0.37
LSTM(2 layers, fixed embedding)	3.02	0.38
biLSTM(1 layer)	2.99	0.38
biLSTM(1 layer, LN)	2.33	0.48
biLSTM(1 layer, GCN via time-step)	2.54	0.46
biLSTM(1 layer, LN, MP)(ours)	1.52	0.57

5 总结

在本次实验中，我们分别在 cifar-10，STSbenchmark 数据集上做了图像分类与文本相似度预测任务，我们给出了一些数据处理与模型结构上的尝试，并取得了一定的提升。

6 组员分工

我们均单独实现了一份 CNN 与 RNN 代码，报告中展示的是黄杨峻实现的 CNN 模型以及冯家苇实现的 RNN 模型。四份代码可见压缩包文件。

表 5: 组员分工

姓名	实验部分	报告部分
冯家苇	RNN、CNN 模型实现	RNN 方法、实验部分撰写 图片设计，报告排版
黄杨峻	CNN、RNN 模型实现 参数可视化实现	CNN 方法、实验部分撰写 PPT 设计，报告排版

参考文献

[1] Sergey Ioffe Jonathon Shlens Zbigniew Wojna Christian Szegedy, Vincent Vanhoucke. Rethinking the inception architecture for computer vision. https://arxiv.org/abs/1512.00567, 2015.

[2] Arizona Phoenix. Siamese recurrent architectures for learning sentence similarity. https://dl.acm.org/citation.cfm?id=3016291, 2016.

[3] pytorch 官方文档. https://pytorch.org/docs/stable/index.html.

[4] Yann LeCun S Chopra, R Hadsell. Learninga similarity metric discriminatively, withapplication to face verification. http://yann.lecun.com/exdb/publis/pdf/chopra-05.pdf, 2005.

[5] Mona Diab Daniel Cera. Semeval-2017 task 1: Semantic textual similarity multilingual and cross-lingual focused evaluation. https://www.aclweb.org/anthology/S17-2001.pdf, 2017.