清华大学电子工程系

媒体与认知 课堂 2

2021-2022 学年春季学期

作业 6

孙一 2022 年 6 月 4 日

调研报告部分

深度学习在应对新冠疫情中的应用——以肺炎 CT 图像检测为例

1 背景调研

自 2019 年 12 月新冠疫情爆发以来，人类社会的正常秩序遭受了巨大冲击，为了遏制疫情的发展，各国政府采用了很多不同的措施。其中，不管是控制疫情扩散还是临床诊断，深度学习技术都得到了广泛的应用Al-Shoqran et al. (2021)。而随着感染人数的持续增长，再加上复杂多变的防疫形势，如何快速、准确地判定受试者是否感染成为了一个重要课题。核酸检测所需的时间往往较长，而肺部 CT 影像也是诊断新冠肺炎的重要依据，但需要人工辨识，因此如何快速地根据 CT 图像准确识别是否感染新冠肺炎成为一个重要的研究方向费宏运 等 (2021)。

2 现有解决措施

采用深度学习领域的图像分类模型可以高效地解决上述问题。比如费宏运 等 (2021) 中加载预训练后的 AlexNet 模型，识别新冠肺炎 CT 图像与非新冠肺炎 CT 图像；王健庆 等 (2020) 中考察了 GoogleNet 和 ResNet 这两种卷积神经模型的识别效果；Ji et al. (2020) 中采用了图像增广、特征融合以及模型的微调方法；Zhang et al. (2021) 中采用了图像切片 + vision transformer 的方法；Krishna et al. (2021) 中采用了卷积神经网络 +Gabor 滤波器的方法。

3 技术效果分析

费宏运 等 (2021) 发现在小数据集上 AlexNet 的效果比较好，但网络深度不够深，而且数据集较少，模型没有较强的鲁棒性；王健庆 等 (2020) 发现 GoogleNet 效果更好，并且发现加载预训练参数可以进一步压缩训练

时间、提升识别准确率；Ji et al. (2020) 中提出的图像增广方法有效地解决了数据不足的问题，并且在特征融合与模型微调的基础上得到了很好的测试准确率；Zhang et al. (2021) 中提出的 UNet 图像分割网络 + Swin transformer 的架构也得到了比较好的结果，表明图像识别领域的注意力机制也可以用于新冠肺炎 CT 图像的识别；Krishna et al. (2021) 中的实验表明，与 VGG16,VGG19, ResNet50, Mobile Net 网络相比，作者提出的 "Novel" 卷积神经网络在 Gabor 滤波器的配合下可以获得最好的识别效果，因为 Gabor 线性滤波器可以去除数据中的噪声。

4 未来优化思路

首先，新冠肺炎 CT 图像有其特殊性，目前仍然缺乏大量、高质量的数据，导致训练的结果会出现过拟合和鲁棒性不佳的情况，因此训练结果不具备在临床条件下的普遍适用性，而建设这样庞大的医疗图像数据库将会面临文化和成本两个方面的严峻挑战。未来可以通过医疗数据中心和数字化医院的建设，以及电子病历等临床数据的补充来解决这一问题王健庆等 (2020)。

其次，深度学习训练模型可以说是一个黑盒，难以结合医学知识合理解释这套系统的技术理论和运行逻辑，在医疗领域难以普及这样的诊断系统。未来可以通过确定区域试实行的方法，将深度学习诊断与医生的个人诊断相结合，逐渐增强患者对深度学习诊断的信任度王健庆 等 (2020)。

另外，由于肺炎 CT 图像标注比较耗时，未来半监督或无监督的识别技术还有待进一步探索Rathod et al. (2021)。

编程部分

5 编程作业报告

5.1 完成 Transformer 场景文本识别任务的程序代码

5.1.1 TODO 1 完成整体模型的初始化

```
# ========================================
# TODO 1: complete the initialization for self.transformer
# ========================================
self.transformer = TransformerModel(
    d_input = 64,
```

```
    d_model = 32,
    n_head = 2,
    num_encoder_layers = 1,
    num_decoder_layers = 1 ,
    dim_feedforward = 32,
    output_class = 40,
    max_timestep=30)
```

5.1.2 TODO 2 完成模型的的前向计算过程

feats: [length = 32, batch_size(b) = 32, feat_size = 64]
tgt: [max_len, b]
tgt_length: [b]
logits: [max_len, b, num_classes = 40]

```
# ========================================
# TODO 2: complete network forward process
# ========================================
logits = self.transformer.forward(feats, tgt, tgt_length)
return logits
```

5.1.3 TODO 3 完成模型的推理过程

preds: [max_timestep = 30, b = 32]
logits: [max_timestep - 1 = 29, b = 32, num_classes = 40]

```
# ========================================
# TODO 3: complete network inference process
# ========================================
preds, logits = self.transformer.inference(feats)
return preds, logits
```

5.1.4 TODO 4 完成位置编码的前向计算过程

```
# apply self.transformer on feature sequences to get the logits
# -- hint: firstly add x with self.pe, and then apply self.dropout to it
# 先添加位置编码然后再做 Dropout, 只需要用到 self.pe 的前 32 位
# ========================================
# TODO 4: complete positional encoding forward process
# ========================================
```

```
x_hat = self.dropout(x + self.pe[:x.size(0), :])
# 注意是 "+" 号
return x_hat
```

5.1.5 TODO 5 完成 Transformer 各模块的初始化

```
# ========================================
# TODO 5: complete the initialization for the transformer model
# ========================================
self.input_fc = nn.Linear(d_input, d_model)

encoder_layer = nn.TransformerEncoderLayer(d_model, n_head, dim_feedforward)
self.encoder = nn.TransformerEncoder(encoder_layer, num_encoder_layers)
decoder_layer = nn.TransformerDecoderLayer(d_model, n_head, dim_feedforward)
self.decoder = nn.TransformerDecoder(decoder_layer, num_decoder_layers)

self.fc = nn.Linear(d_model, output_class)
self.max_timestep = max_timestep
```

5.1.6 TODO 6 完成 Transformer 的推理过程

```
# ========================================
# TODO 6: complete the inference process for the transformer
# ========================================
src = self.pos_emb(self.input_fc(src))  # 先过一个线性层，然后做位置编码
memory = self.encoder(src)  # 前传获得 memory

tgt = torch.zeros(self.max_timestep, src.size(1)).long().to(src.device)
# 构建目标 tgt: [max_timestep, batch_size]，现在元素全是 0，即 <sos>
for t in range(1, self.max_timestep):
# 推理阶段是串行的，和 RNN 类似，故需要按时间循环
    tgt_emb = self.pos_emb(self.char_emb(tgt[:t,:]))
    # 预测 t 时刻输出时，只需把前 0 到 t-1 的 tgt 送进来就行了，这样可以减少运算
    tgt_mask = self._generate_square_subsequent_mask(t).to(src.device)
    # 相应的 tgt_mask 只需要 t*t，上三角矩阵（右上方是-inf）
    decoder_output = self.decoder(tgt_emb, memory, tgt_mask = tgt_mask)
    # tgt_mask 的大小决定 decoder_output 的大小
    char_cls = self.fc(decoder_output[-1, ...])
```

```
    char_cls = torch.argmax(char_cls, dim = 1)  # 选择预测概率最大元素的 index
    tgt[t, :] = char_cls # 获得第 t 时刻的预测输出
logits = self.fc(decoder_output)
return tgt, logits
```

5.1.7 TODO 7 完成整体网络的的训练过程

```
# ==================================
# TODO 7: complete train_one_epoch()
# ==================================
model.train()
total_loss = 0.0
for step, (ims, words) in enumerate(trainloader):
    tgt, tgt_length = label_converter.encode(words)
    ims, tgt, tgt_length = ims.to(device), tgt.to(device), tgt_length.to(device)
    logits = model(ims, tgt, tgt_length)
    log_probs = nn.functional.log_softmax(logits, dim = 2)
    log_probs = log_probs[:-1, :, :].view(-1, log_probs.size(2))
    # 需要去掉 log_probs 的最后一行 (为了与 target 的长度一致), 并且将前两维压缩为一维
    target = tgt[1:, :].view(-1)
    # 需要去掉 target 的第一行, 因为第一行是 <sos>, 解码时不会出现 <sos>
    # log_probs 和 target 的形状为 [~, 32, 40]
    # 第一个维度取决于 label 中最长的那个的长度 * 32
    loss = criterion(log_probs, target)
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    total_loss += loss.item()
return total_loss / len(trainloader)
```

5.1.8 验证代码正确性

```
(base) E:\2022_1\MR\Media_Cognition_hw\hw6>python network.py
The output size of model is correct!
```

5.1.9 分析 Transformer 训练时的并行特性，推理时与 CTC 解码的区别

首先注意一点，Transformer 训练时的并行特性指的是解码器的并行特性，编码器在训练阶段和推理阶段都是“并行”的，即编码器会关注整个输入序列的所有单词，帮助模型对当前单词更好地进行编码。解码器的作用是利用编码器的输出（作为 K 和 V）和**已经解码出**的单词（经过 Masked Multi-Head Attention 后作为 Q），预测下一个单词（其实是给出下一个词在词汇表中的概率分布）。
在训练阶段，文本真值已知，所以可以代替已解码结果，将本批次的真值送入“引入注意力系数掩码的多头注意力层，该层有一个-inf 的上三角掩码矩阵，用来限制在解码 t 时刻的输出时，只会用到 1 到 t-1 时刻的样本真值。而 Transformer 在推理解码时，由于没有真值，所以和 RNN 类似，只能利用之前时刻**已经解码出**的单词，因此是严格按照时序的串行解码，并且网络需要用到 1 到 t-1 时刻的所有输出，RNN 只需要用到 t-1 时刻的输出，因为 Transformer 没有 RNN 那样的内部状态。故推理阶段 Transformer 不具有并行特性。
CTC 解码的过程是对前面循环神经网络或者 LSTM 输出的结果做逐帧预测，对得到的解码路径做合并相邻重复字符、去除“空白符号”操作进而输出字符序列。

5.2 训练、预测、可视化

5.2.1 训练

模型在训练集上的 loss 和验证集上的单词识别正确率（即完全识别正确的图像样本所占总样本的比例）变化情况见图 1，验证集上的最终正确率为:

```
Epoch [100/100] start ...
train loss = 0.929, validation word accuracy = 43.6%
[Info] model saved in models\model_epoch100.pth
loss and accuracy curves has been saved in loss_and_accuracy.jpg
```

由于作业侧重于理解原理，数据量较小，在使用默认网络参数的条件下，训练 100 轮后在验证集上的单词识别正确率约为 50%，与 HW5 相差不多，而且正确率在训练后期的相对变化已经很小，说明此时模型在训练集上已经取得了比较稳定的结果。要想提高模型在验证集上的准确率，可以通过增加训练样本或者适当提高模型的复杂度来实现。
同时可以发现，与 HW5 的 CRNN+CTC 识别的结果相比，transformer 在验证集上的正确率波动更大；另外，在 batch_size 相同的情况下，考虑到

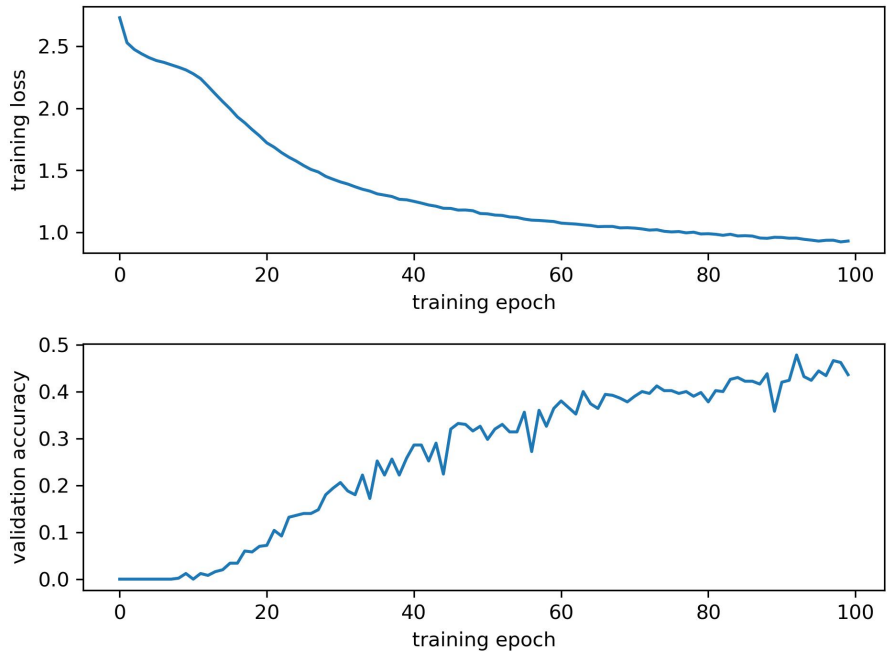

图 1: loss_and_accuracy

Transformer 的复杂度更高，训练 epoch 数也更多，而训练时长与 HW5 相差不多，充分说明 Transformer 训练时的并行特性可以加快训练速度。

5.2.2 使用训练好的模型预测新的文本图像

默认图片的识别结果：

```
prediction: parking
CTC visualization has been saved as data/my_own/a_vis.jpg
```

可视化结果如图 2。从图中可以看出：

1. 因为 transformer 在编码当前字母时可以关注到整个输入序列中的所有字母，所以每一个输出的长度和图片中对应字母的长度接近，说明 transformer 在这段时间可以联系上下文信息进而持续输出相同的结果，不会像 HW5 中一样，在字母发生突变的边界输出一小段识别结果，然后是一段较长的 blank 符号 “-”。
2. transformer 的识别结果比图片中的字母要早大概一个字母。

自选图片 1 识别结果：

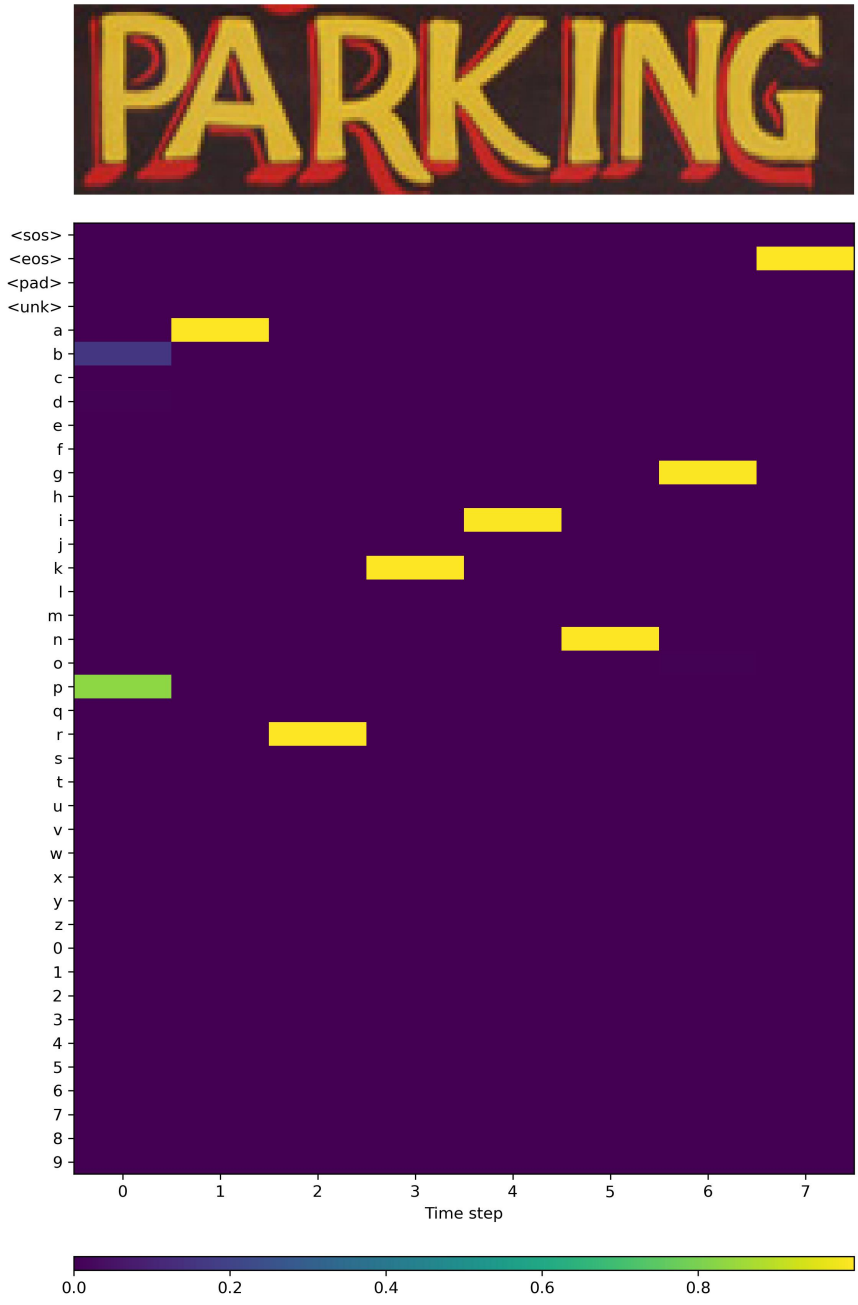

图 2: parking_vis

```
prediction: marvel
CTC visualization has been saved as data/my_own/marvel_vis.jpg
```

可视化结果如图 3。与默认图片情形相似。

自选图片 2 识别结果：

```
prediction: csmoeowe
CTC visualization has been saved as data/my_own/casino_royal_vis.jpg
```

可视化结果如图 4。此时由于字母间距较小，识别错误率大大增加。

自选图片 3 识别结果：

```
prediction: mrastumis
CTC visualization has been saved as data/my_own/marvelstudios_vis.jpg
```

可视化结果如图 5。与图 3相比，此图多了一个单词”STUDIOS”，”MARVEL” 在图片中的占比明显下降，而且”STUDIOS” 上下还有两条横线作为干扰，因此识别准确率有所下降。

此外，还可以观察到：字母”e” 和”s” 一般不会识别错误，可能是因为英文单词中字母”e” 和”s” 出现的概率非常高，等效于增大了”e” 和”s” 的训练样本数，因此识别准确率相对更高一些。

参考文献

Al-Shoqran M, Shorman S. Applications of artificial intelligence in covid-19 pandemic[C]//European, Asian, Middle Eastern, North African Conference on Management & Information Systems. Springer, 2021: 313-321.

费宏运, 陈庚, 迟兆瑞, 等. 基于 AlexNet 的新冠肺炎 CT 图像识别的可行性分析[J]. 信息与电脑 (理论版), 2021, 33: 137-140.

王健庆, 王琪🄴. 基于深度学习的新冠肺炎 CT 图像识别研究[J]. 信息与电脑, 2020, 32(17): 62-64.

Ji D, Zhao Y, Zhang Z, et al. Research on recognition method of covid-19 images based on deep learning[J]. medRxiv, 2020.

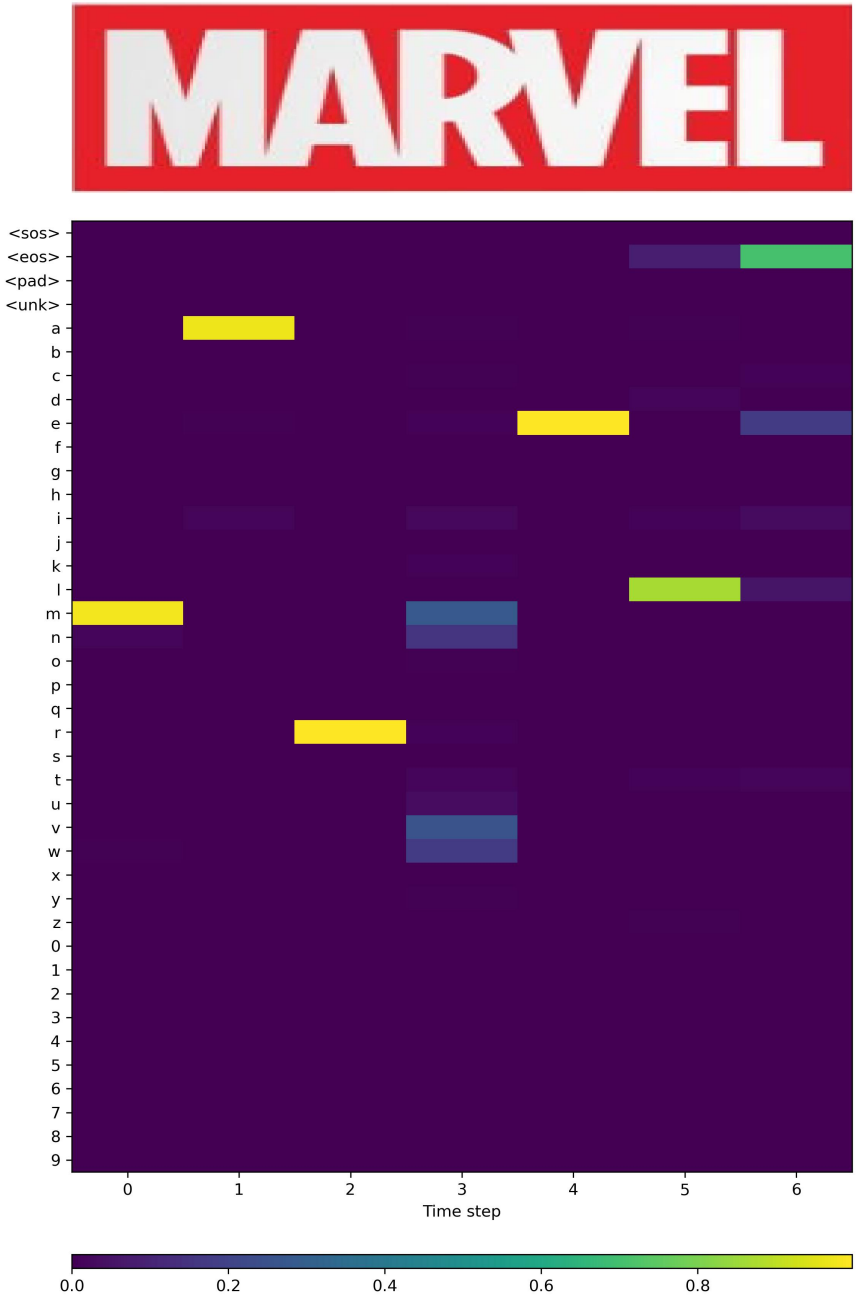

图 3: marvel_vis

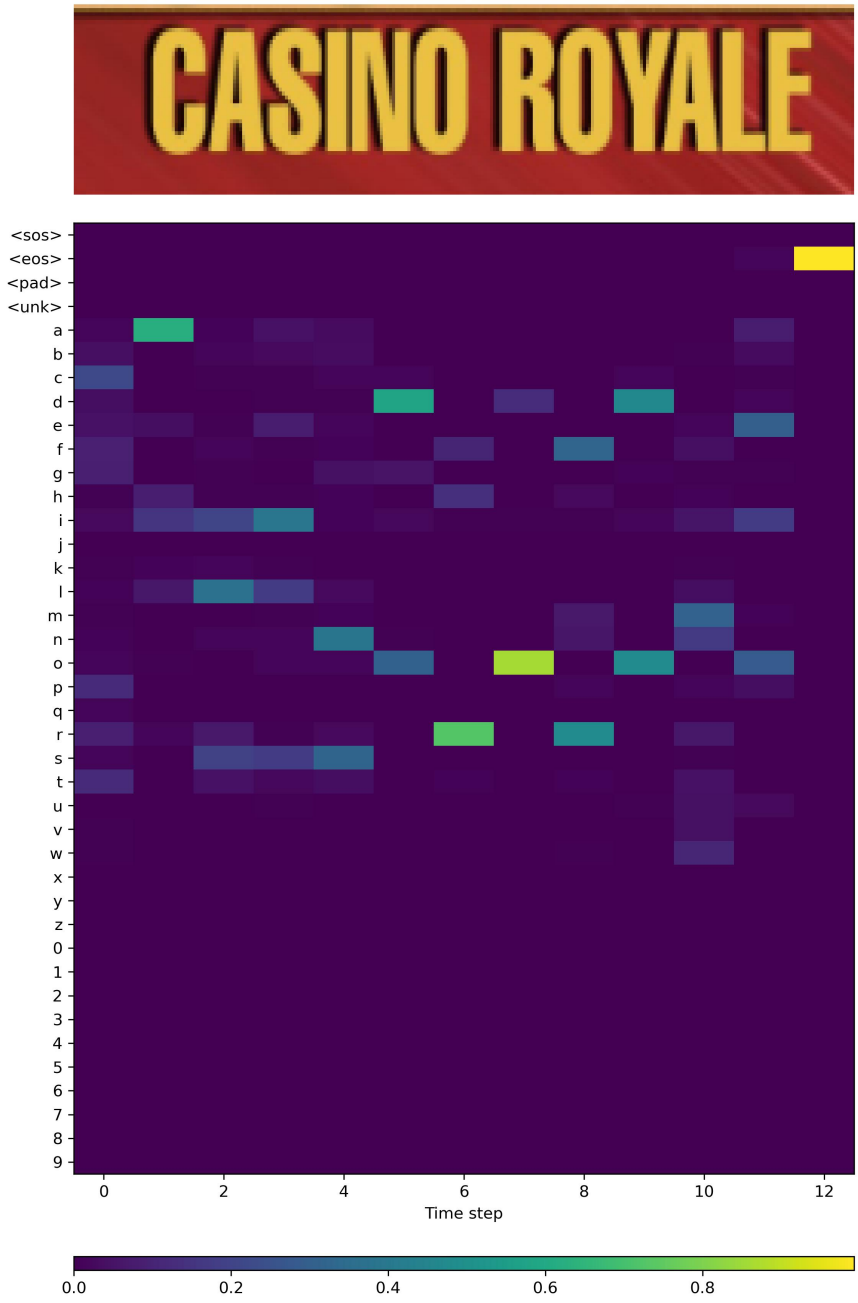

图 4: casino_royal_vis

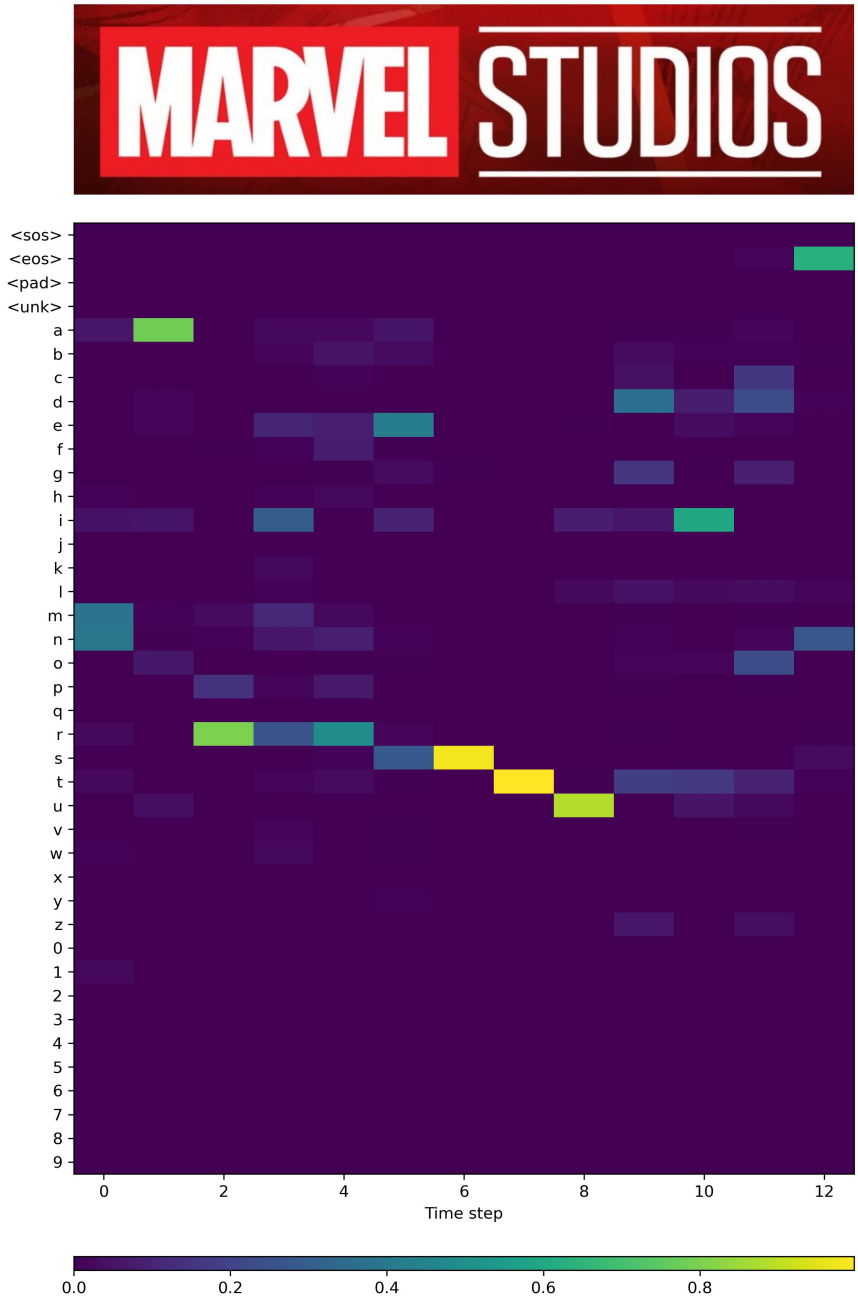

图 5: marvel_studios_vis

Zhang L, Wen Y. A transformer-based framework for automatic covid19 diagnosis in chest cts[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 513-518.

Krishna B V, Bodavarapu P N R, Santhosh P, et al. Chest computed tomography scan images for classification of coronavirus by enhanced convolutional neural network and gabor filter[C]//2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS). IEEE, 2021: 825-831.

Rathod S R, Khanuja H K. Automatic segmentation of covid-19 pneumonia lesions and its classification from ct images: A survey[C]// 2021 International Conference on Intelligent Technologies (CONIT). IEEE, 2021: 1-8.