TIME SERIES GENERATION WITH MASKED AUTOENCODER

Mengyue Zha; Mengyue Zha; Mengqi Liu; Tong Zhang; Kani Chen;

摘要

本文表明带外推器的掩码自动编码器 (ExtraMAE) 是一种用于时间序列生成的可扩展自监督模型。ExtraMAE 随机屏蔽原始时间序列的一些补丁，并通过恢复屏蔽的补丁来学习时间动态。我们的方法有两个核心设计。首先，ExtraMAE 是自我监督的。监督允许 ExtraMAE 有效且高效地捕获原始时间序列的时间动态。其次，ExtraMAE 提出了一个外推器来解开解码器的两个工作：恢复潜在表示并将它们映射回特征空间。这些独特的设计使 ExtraMAE 在时间序列生成方面始终如一地显着优于最先进的 (SoTA) 基准。轻量级架构还使 ExtraMAE 快速且可扩展。ExtraMAE 在时间序列分类、预测和插补等各种下游任务中表现出色。作为一种自我监督的生成模型，ExtraMAE 允许对合成数据进行显式管理。我们希望本文将开创一个使用自监督模型生成时间序列的新时代。

关键词：时间序列；自监督模型；ExtraMAE；轻量级架构

1 引言

在深度学习时代，模型的训练往往以来高质量数据的充足供应，但由于许多领域比如医疗等无法提供合格或包含敏感信息的数据，对数据的需求已经成为时间序列模型的瓶颈。因此可以考虑数据的合成，好的合成数据尊重原始数据的时间动态，在实际应用中可以替代原始数据。

2 相关工作

2.1 基于生成式的序列数据重构方法

无监督生成对抗网络 (GAN)[1]。可以用于生成合成数据。GAN 学习从随机噪声到原始数据分布的映射。理论上，GAN 可以通过从随机噪声中采样产生任意数量的样本。C-RNN-GAN[2]是首次尝试用 GAN 生成合成时间序列，C-RNN-GAN 通过循环神经实现生成器和鉴别器。

RCGAN[3]在 C-RNN-GAN 基础上增加附加条件信息，生成器和鉴别器都以辅助信息为条件。目前大量 GAN 变体生成特定领域的合成数据，如金融[4]、传感器[5]和决策[6]等。

2.2 包含时间特性的序列数据重构方法

序列数据的时间动态是数据生成的核心，而上述 GAN 的变体模型都没有考虑时间序列数据的时间性质。为了更好地捕获时间序列数据中的时间依赖性，TimeGAN[7]在潜在空间中设计了一个监督预测任务。目前已知 TimeGAN 是唯一一个成功保存原始数据时间动态的时间序列生成模型。尽管有监督训练在 TimeGAN 中取得了成功，但无监督模型仍然主导着时间序列生成。

2.3 有/无监督生成模型的区别

1. 训练难度不同。监督生成模型很少遇到在非监督生成模型中的问题，一些典型的问题如不收敛，梯度消失，模式崩溃[8]。

2. 可操作性不同。无监督生成模型与无关的操作不兼容，其合成数据对下游任务是盲目的，不能管理合成数据的多样性，也无法估算缺失的价值 (见第 3 节)；GAN 将随机噪声作为生成的种子，无法系统地管理生成。然而，监督生成模型允许我们对生成进行很好的控制，可以直接确定合成数据的多样性。

3 本文方法

3.1 模型结构

基于上述相关工作的论述，论文提出了一种时间序列生成监督模型———带外推器的蒙面自编码器 (ExtraMAE)。自监督训练时，每次迭代 ExtraMAE 会随机屏蔽输入时间序列的一些片段，并从未屏蔽的片段中推断缺失的片段。ExtraMAE 提出的外推器放弃了掩码标记，从而避免了合成时间序列的不连续 (参见补充材料)。论文修剪了堆叠的 Transformer 块[9]，这在以前的基于掩码的模型中非常流行[10-12]。采用轻量级的 RNN 替换了沉重的 Transformer 块，提高了模型的训练速度。ExtraMAE 主要分为三个部分：编码器、外推器和解码器，模型结构如图 1所示：

编码器（Encoder）：由两层 RNN 网络构成，用于对切分、打乱、遮盖后的序列片段进行编码。

外推器（Extrapolator）：由两层全连接网络构成，用于学习编码后数据之间的依赖性，并恢复被遮盖部分的数据。

解码器（Decoder）：结构同编码器部分一样，用于对经过外推器恢复的数据进行解码。

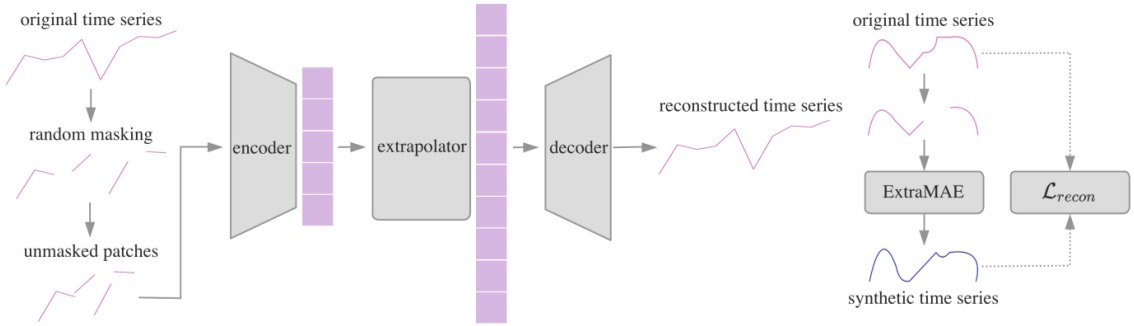


图 1: 模型结构

3.2 损失函数定义

论文采用重构数据与原始数据之间的平均均方误差（MSE）作为模型训练的损失函数，公式如下:

$$L_{recon} = \mathbb{E}_{\mathrm{x}} \left\| \mathrm{X} - \hat{\mathrm{X}} \right\|^2 \tag{1}$$

3.3 模型测试

测试环节包含四种测试方法，分别是随机遮盖片段重构全部序列、多次重复前一种方法并取平均值、依次遮盖序列全部并拼接重构的片段（类似交叉验证）、多次重复前一种方法并取平均值，其中片段拼接示意图如图 1。除了可视化结果，论文还引入了区别分数和准确分数用以展示模型的效果。

区别分数是从重构数据与原始数据尽可能相似的角度，在原始数据和重构数据上训练一个 2 层 LSTM 网络作为事后鉴别器，原始数据标签为 1，重构数据标签为 0，鉴别器的分类准确率为区别分数。若重构数据与原始数据相似，则鉴别器不能很好地区分原始/重构数据，即区别分数低，模型效果较好。

准确分数是从重构数据可以很好的代替原始数据的角度，在重构数据上训练一个 2 层 LSTM 网络

作为预测器，预测分数是预测器在原始数据上的平均绝对测试误差（MAE）, 示意图如图 3。若重构数据可以对原始数据做出准确的预测，即较低的预测分数表明重构数据与原始数据相似度很高，模型效果较好。

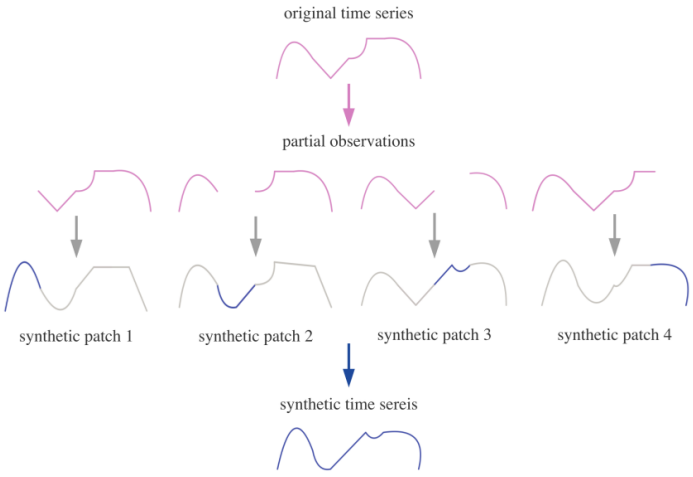


图 2: 片段拼接

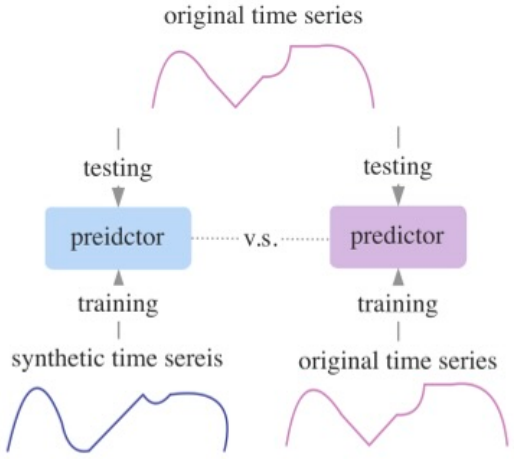


图 3: TSTR 实验

4 复现细节

复现的论文已有源码，主要展示源码模型部分和自己添加的部分

4.1 模型代码

模型基于 Pytorch 框架搭建，编码器和解码器都是由两层 RNN 构成，外推器由两层全连接网络构成，网络层数等参数在相应数据配置文件中给出。

```
# 编码器（Encoder）
class Encoder(nn.Module):
    # 升维
    def __init__(self, args):
        super(Encoder, self).__init__()
        self.rnn = nn.RNN(input_size=args.z_dim,
                          hidden_size=args.hidden_dim,
                          num_layers=args.num_layer)
        self.fc = nn.Linear(in_features=args.hidden_dim,
                            out_features=args.hidden_dim)

    def forward(self, x):
        x_enc, _ = self.rnn(x)
        x_enc = self.fc(x_enc)
        return x_enc


# 解码器（Decoder）
class Decoder(nn.Module):
    # 降维
    def __init__(self, args):
        super(Decoder, self).__init__()
        self.rnn = nn.RNN(input_size=args.hidden_dim,
                          hidden_size=args.hidden_dim,
                          num_layers=args.num_layer)
        self.fc = nn.Linear(in_features=args.hidden_dim,
                            out_features=args.z_dim)

    def forward(self, x_enc):
        x_dec, _ = self.rnn(x_enc)
```

```
        x_dec = self.fc(x_dec)
        return x_dec

# 外推器（Extrapolator）
class Interpolator(nn.Module):
    def __init__(self, args):
        super(Interpolator, self).__init__()
        self.sequence_inter = nn.Linear(in_features(args.ts_size - args.
           total_mask_size),out_features=args.ts_size)
        self.feature_inter = nn.Linear(in_features=args.hidden_dim,
                                       out_features=args.hidden_dim)

    def forward(self, x):
        # x(bs, vis_size, hidden_dim)
        x = rearrange(x, 'b l f -> b f l')
        x = self.sequence_inter(x)
        x = rearrange(x, 'b f l -> b l f')
        x = self.feature_inter(x)
        return x
```

4.2 我的工作

主要添加了 Argo 数据可视化前恢复切分数据为原始序列顺序的部分，采取类似滑动窗口的方法。由于最初切分时滑动窗口为 1，片段之间存在重复数据，因此在恢复时根据切分的片段长度，选取对应位置开始的连续片段进行拼接，恢复成原始顺序，以尽可能保证数据之间依赖性。

```
# 获取数据长度
def get_length():
    osimples_length = np.loadtxt('data/ArgoLength_MAE.csv', delimiter=',',
       skiprows=1).tolist()
    simples_length = []
    for i in range(len(osimples_length)):
        simples_length.append(int(osimples_length[i]))
    return simples_length

# 根据长度信息拼接片段数据
def merge_data(sdata, simples_length):
    datas = []
    j, l, flag = 0, 0, 1
    for s in range(100):
        data = []
        simple_length = simples_length[s]
        if s == 0:
            for j in range(sdata[s].shape[0]):
                data.append(sdata[s][j])
            j = j + 1
        dl = len(data)
        if flag == 0:
            j = j + 24
        while dl < simple_length:
            for k in range(l, sdata[j].shape[0]):
                data.append(sdata[j][k])
                dl += 1
                if dl == simple_length:
                    if k < sdata[j].shape[0] - 1:
                        l = k + 1
                    else:
                        l ,flag= 0, 0
                    break
            if dl < simple_length:
```

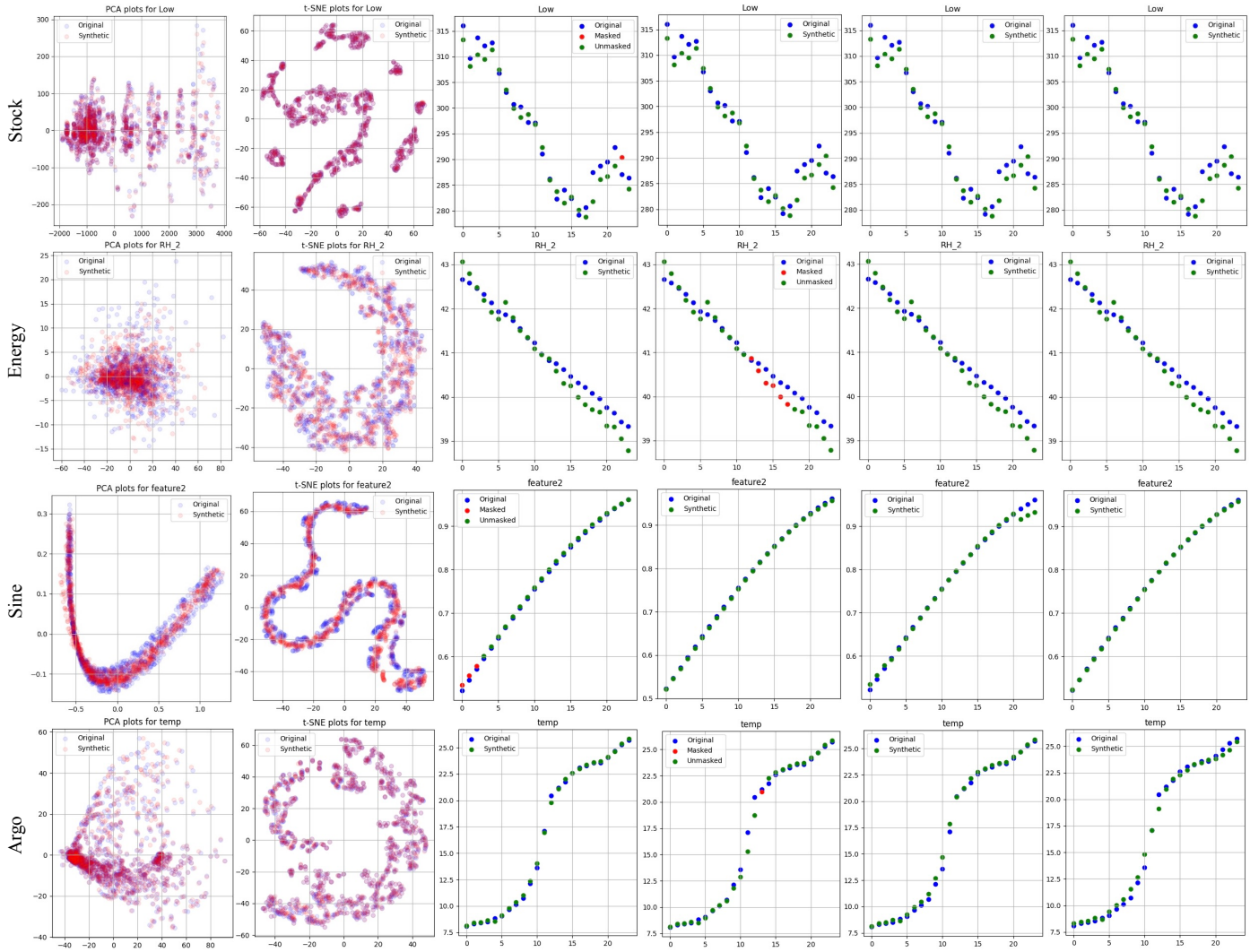
```
                l = 0
                j = j + 24
        data = np.array(data)
        datas.append(data)
    return datas

# 获取拼接后的数据
def get_merge_data(data):
    length = get_length()
    return merge_data(data, length)
```

5 实验结果分析

实验主要分为两个部分，第一部分是基于原论文在四种数据上的实验，第二部分是针对不打乱切割后的 Argo 数据的实验。

5.1 基于原论文的四种数据的实验结果

首先对论文中的股票、能源和正弦三种数据，以及 Argo 数据进行模型训练和测试。复现采用论文中最佳效果参数，并与论文中结果进行比对。对于四种数据，由于四种测试方法在原始/重构数据降维后的结果类似且图片较多，所以此次对于每种数据，只选择同一序列片段中同一个特征在经过四种测试方法得到的结果图以及所有数据中该特征经过随机遮盖重构方法得到的两种降维图。可视化结果及测试分数如下：



(s) PCA (t) t-SNE (u) 随机 (v) 随机-平均 (w) 拼接 (x) 拼接-平均

图 4: 测试结果图

表 1: 预测分数

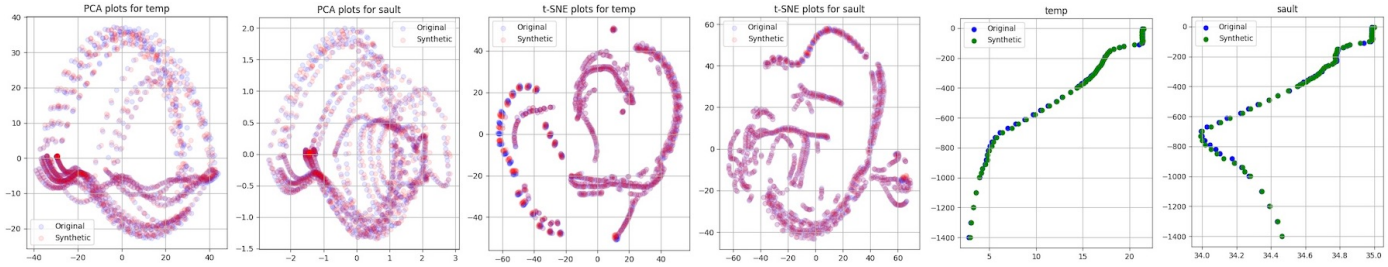
Predictive score	Stock	Energy	Sine	Argo
Random	0.038	0.191	0.286	0.019
Random average	0.038	0.191	0.284	0.019
Cross concatenate	0.037	0.191	0.256	0.018
Cross concatenate average	0.038	0.191	0.280	0.019
Paper Results	0.037±0.000	0.101±0.000	0.256±0.001	

表 2: 区别分数

Discriminate score	Stock	Energy	Sine	Argo
Random	0.063	0.191	0.500	0.045
Random average	0.101	0.037	0.500	0.031
Cross concatenate	0.069	0.093	0.500	0.092
Cross concatenate average	0.100	0.019	0.500	0.064
Paper Results	0.071±0.020	0.019±0.010	0.410±0.107	

5.2 非乱序 Argo 数据实验结果

为了方便可以拼接重构的序列片段，此次对切分后不随机打乱的 Argo 数据也进行进行训练和测试，然后将原始/重构数据的序列片段拼接成原顺序的形式。由于四种方法结果图较多且相似，因此只选择一条完整剖面数据的温度和盐度两个特征经过随机遮盖重构方法后得到的结果图，可视化结果如下：



(a) 温度 PCA (b) 盐度 PCA (c) 温度 t-SNE (d) 盐度 t-SNE (e) 剖面温度 (f) 剖面盐度

图 5: 顺序 Argo 测试结果

上述实验结果表明该模型对于序列数据具有较好的重构效果，复现的结果与论文中结果相差不大，但由于能源数据重构可视化效果未达到理想效，后续可进一步改进优化。此外，Argo 原始数据和重构数据进行原始顺序拼接后的可视化效果良好，可用于后续学习研究，因此可进一步改进优化。

6 总结与展望

此次所复现的论文中提出一种 ExtraMAE 网络结构，用于序列数据的重构，论文选取股票数据、能源数据和正弦数据三种数据进行模型训练，并用四种测试方法进行模型测试，此次所复现的结果与论文中所列结果大体一致，我也学习了解了序列数据生成模型以及模型的设计、验证思路。此外，由于后续学习的需要，本次复现还采用了 Argo 数据进行模型训练和测试，在可视化过程中将最初经过滑动窗口分割的序列数据根据其原有长度再次采用类似简易版滑动窗口的方法进行数据恢复，可视化结果显示模型效果良好。但在此次复现过程中在能源数据上的可视化重构效果不理想，并且 Argo 数据由于序列长度不一，在可视化恢复为原始数据时存在一些问题，在后续可继续尝试调试改进模型，分别解决这两个问题。

参考文献

[1] GOODFELLOW I J, POUGET-ABADIE J, MIRZA M, et al. Generative Adversarial Nets[J]. Advances in Neural Information Processing Systems, 2014, 27.

[2] MOGREN O. C-RNN-GAN: Continuous recurrent neural networks with adversarial training[J]., 2016. arXiv: 1611.09904.

[3] ESTEBAN C, HYLAND S L, RÄTSCH G. Real-valued (medical) time series generation with recurrent conditional gans[J]., 2016.

[4] NI H, SZPRUCH L, WIESE M, et al. Conditional Sig-Wasserstein GANs for Time Series Generation [J]., 2020. arXiv: 2006.05421.

[5] DEBAPRIYA H, YUNG-CHEOL B. Synsiggan: Generative adversarial networks for synthetic biomedical signal generation[J]. Biology, 2020, 09(12): 441.

[6] SUN H, DENG Z, CHEN H, et al. Decision-Aware Conditional GANs for Time Series Data[J]., 2020. arXiv: 2009.12682.

[7] JINSUNG Y, DANIEL J, der SCHAAR MIHAELA V. Time-series Generative Adversarial Networks [J]. Advances in Neural Information Processing Systems, 2019, 32.

[8] WIATRAK M, ALBRECHT S V, NYSTROM A. Stabilizing Generative Adversarial Network Training: A Survey[J]., 2019. arXiv: 1910.00927.

[9] ASHISH V, NOAM S, NIKI P, et al. Attention is All you Need[J]. Advances in Neural Information Processing Systems, 2017, 30.

[10] BAO H, DONG L, WEI F. BEiT: BERT Pre-Training of Image Transformers[J]. ICLR, 2022.

[11] DEVLIN J, CHANG M, LEE K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[J]. NAACL-HLT, 2019.

[12] KAIMING H, XINLEI C, SAINING X, et al. Masked Autoencoders Are Scalable Vision Learners[J]. CVPR, 2022: 15979-15988.