

# simCLRとマルチ学習モデルを使用 した自己教師学習

レポート: 自己教師学習モデル

SimCLRと対照的マルチ学習

VIC登録

著者：ヴェダント・ジェイン

Git-hub リポジトリ リンク

# 私の紹介

名前 - VEDA NT ジエイン

大学 - IIT (ISM) ダンバード

スキル - Python、 C++、 JupyterNotebook、  
Tableau、 Numpy、 パンダ、 Matplotlib、  
シーボーン、 ドッカー およびExcel

趣味 - 好きなこと 時間を投資する

暗号市場と 分析とNFTメタ  
ウェブ3。

「Deepcraft.aiチームの皆様、

このメッセージがあなたのお元気を願っています。

概説された目的に沿ってプロジェクトを完了しました。このプロジェクト

Deepcraft.aiの目標に沿うための私の献身と努力を表しています  
ビジョンと目標。

最終的な成果物は次のとおりです。

- 【チーフAIエンジニアプロジェクトレポート】
- [Google Collab Notebook と Git-hub リンク]

あなたのエキサイティングな活動に貢献する機会をいただけて光栄です  
皆さんのフィードバックをお待ちしています。

このプロジェクトに参加する機会を与えていただき、ありがとうございます。」

# コンテンツの概要 -:

1 自己教師あり学習の背景研究（方法と課題）

2 データセット分析（欠損値のクリーニングと処理）

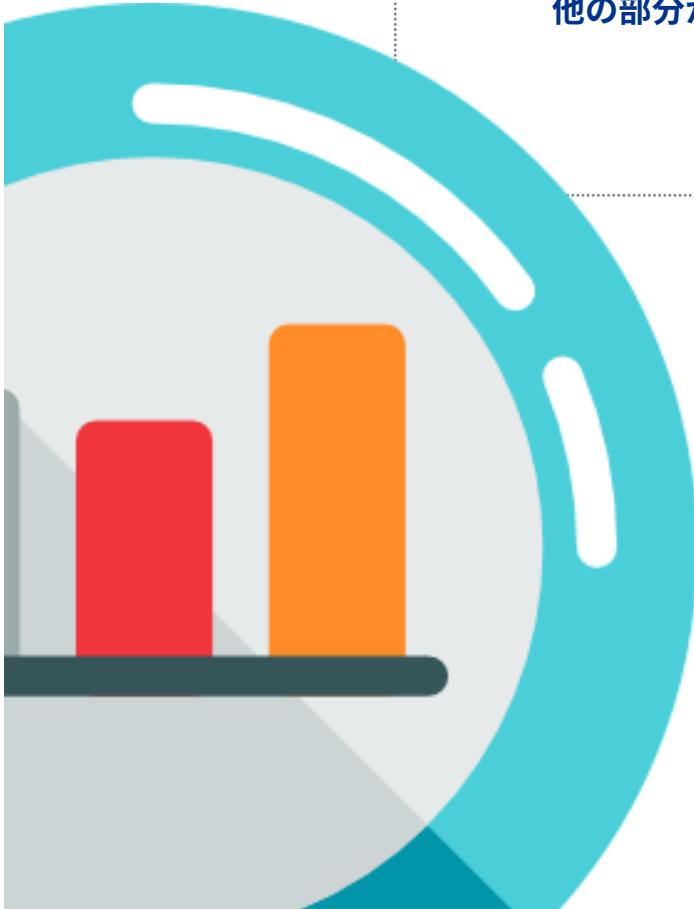
3 技術概要（モデル研究）

4 評価指標

5 検証結果

まとめと今後の展望

# 「SSL とは何ですか??」



自己教師学習は、ラベル付けされたデータに依存せずに、モデルが入力データの一部を他の部分から予測することを学習する機械学習の一種です。

モデルが有用な表現を学習できるようにするタスクを設定することで、データ自体から独自のラベルを生成します。

## 事前タスク 創造



プレテキストタスクは、モデルに不完全または破損した入力バージョンが与えられ、欠落または変更された部分を予測するタスクが課されるように設計されています。これにより、モデルはデータ自体から有用な機能を学習できます。

## 表現学習



モデルは、前提タスクを解決することでデータの表現(特徴)を学習します。この段階では、モデルは特定のラベルには関心がなく、生の入力データからパターン、構造、または特性を学習することに重点を置いています。



## 自己監督 (ラベル生成)

自己監督では、データ自体の構造または一部を使用して疑似ラベルを作成します。これがSSLの重要な部分です。モデルは独自の監督信号を生成します。



## 微調整 (転送) 学ぶ)

口実課題から学習した特徴は、次の課題でのパフォーマンスを向上させるために転送され、適応される。  
下流タスク。

# 直面する課題:



## 効果的な口実タスクの設計

SSL の有効性は、ラベルを生成するために使用される口実タスクに大きく依存します。



## ラベルのない汚れたデータへの恐怖:

教師あり学習に比べて利用可能なデータが少なく、利用可能なデータも理想的にはクリーンではないため、モーデリングの前に、事前の作業、つまりクリーニングと分類を行う必要があります。



## 表現の崩壊と過剰適合:

SSLモデルは表現の崩壊に悩まされる可能性があり、学習した表現が区別されなくなる。サンプル間での差異が小さいため、下流のタスクでのモデルの有効性が低下します。



## 最適化と安定性の問題

明確に定義されたラベルがないと、モデルの進捗状況を評価することが難しくなり、安定性。



## 実際のアプリケーションへの移植性:

SSL方式は研究の場ではうまく機能することが多いが、現実世界に適用するには困難を伴う。多様な、ノイズの多い、または動的なデータを扱う実稼働レベルのアプリケーション。

# 報告書は以下に焦点を当てます

SimCLR(対照学習)とVICReg(マルチインスタンス学習)

## シムCLR: (CL)

$$l_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{k \neq i} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \quad (1)$$

where  $\text{sim}(\cdot)$  represents the cosine similarity and  $\mathbb{1}_{k \neq i} \in [0, 1]$  corresponds to 1 if  $k \neq i$ , otherwise 0. The loss is further tunable with an additional temperature parameter  $\tau$ . The final overall loss is then composed of all positive pairs

## バッチペア機能

## アルゴリズム

### Algorithm 1: SimCLR-TS Algorithm

```
Input: labeled or partially labeled Dataset  
        $\mathcal{D} = \mathcal{D}_{\text{labeled}} \cup \mathcal{D}_{\text{unlabeled}}$ , Set of Augmentations  $\mathcal{A}$ ,  
       models  $f_\psi(\cdot), g_\phi(\cdot)$ , contrastive loss  $l(\cdot)$ , cross-entropy  
       loss  $XE(\cdot)$   
for epoch in  $\text{epochs}_{\text{cl}}$  do  
    for batch in  $\mathcal{D}$  do  
       $\tilde{x}_i, \tilde{x}_j = \text{augment}(\text{batch})$ ,  $\text{augment}(\cdot) \in \mathcal{A}$   
       $h_i = f_\psi(\tilde{x}_i)$   
       $h_j = f_\psi(\tilde{x}_j)$   
       $\psi = \psi + \nabla l(h_i, h_j)$   
    end  
end  
 $f(\cdot) = \text{freeze\_weights}(f_\psi(\cdot))$   
for epoch in  $\text{epochs}_{\text{classifier}}$  do  
    for batch, label in  $\mathcal{D}_{\text{labeled}}$  do  
       $y_i = g_\phi(f(\text{batch}))$   
       $\phi = \phi + \nabla XE(y_i, \text{label}_i)$   
    end  
end
```

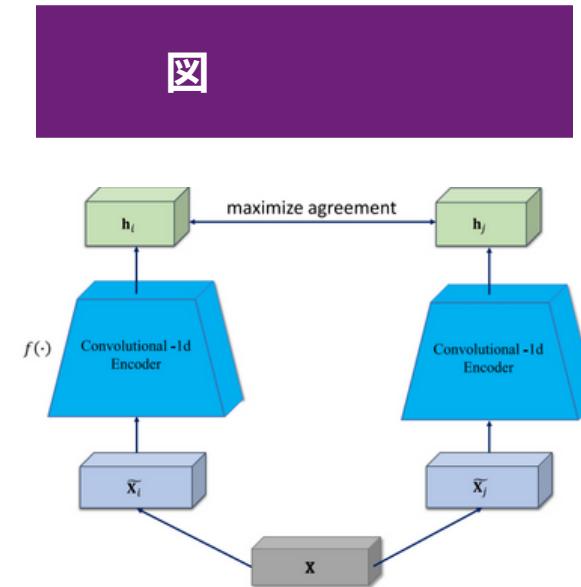


Fig. 2. SimCLR-TS structure.

# 報告書は以下に焦点を当てます

SimCLR(対照学習)とVICReg(マルチインスタンス学習)

## VIReg: (ミル)

### バッチペア機能

$$s(Z^A, Z^B) = \frac{1}{n} \sum_{b=1}^n \|z_b^A - z_b^B\|_2^2. \quad (14)$$

In addition, the covariance criterion  $c(Z)$  in VICReg is defined as

$$c(Z) = \frac{1}{d} \sum_{i \neq j} [C(Z)]_{i,j}^2, \quad (15)$$

where  $C(Z)$  represents the covariance matrix of  $Z$ . The overall loss of VICReg is a weighted sum of the variance, invariance, and covariance:

$$\mathcal{L} = s(Z^A, Z^B) + \alpha(v(Z^A) + v(Z^B)) + \beta(C(Z^A) + C(Z^B)) \quad (16)$$

$$v(Z^A) = \frac{1}{d} \sum_{j=1}^d \max(0, \gamma - S(z_j^A, \varepsilon)). \quad (12)$$

Here,  $z_j^A$  represents the vector composed of each value at dimension  $j$  in  $Z^A$  and  $S$  represents the regularized standard deviation, defined as

$$S(y, \varepsilon) = \sqrt{\text{Var}(y) + \varepsilon}. \quad (13)$$

### バーロウ・ツインズ

Barlow Twinsは、例の歪んだバージョンから埋め込みベクトルの類似性を促進しながら、再帰を最小化する新しい損失関数を導入しました。

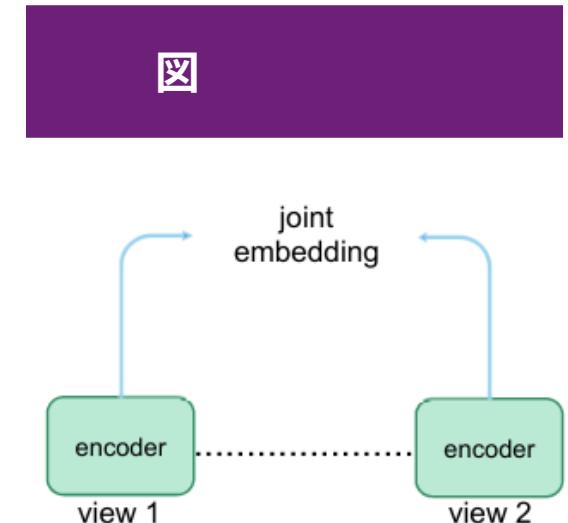
コンポーネント間の冗長性。

$Z^A$  and  $Z^B$ . The loss function of Barlow Twins is defined as

$$\mathcal{L}_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2. \quad (10)$$

Here,  $\lambda$  is a hyper-parameter, and  $C$  represents the cross-correlation matrix computed between the two batches of embeddings  $Z^A$  and  $Z^B$ , defined as

$$C_{ij} = \frac{\sum_b z_{b,i}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2} \sqrt{\sum_b (z_{b,j}^B)^2}}, \quad (11)$$



# 事前タスク??

## データセットの特性

### コンテキストベースの事前タスク

コンテキストベースの方法は、提供された例間の固有のコンテキスト関係に依存し、空間構造やローカルとグローバルの両方の一貫性の維持などの側面を網羅します。

データを操作して、モデルのトレーニングでそれがどのように機能するかを理解します。

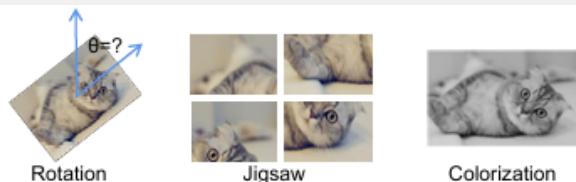
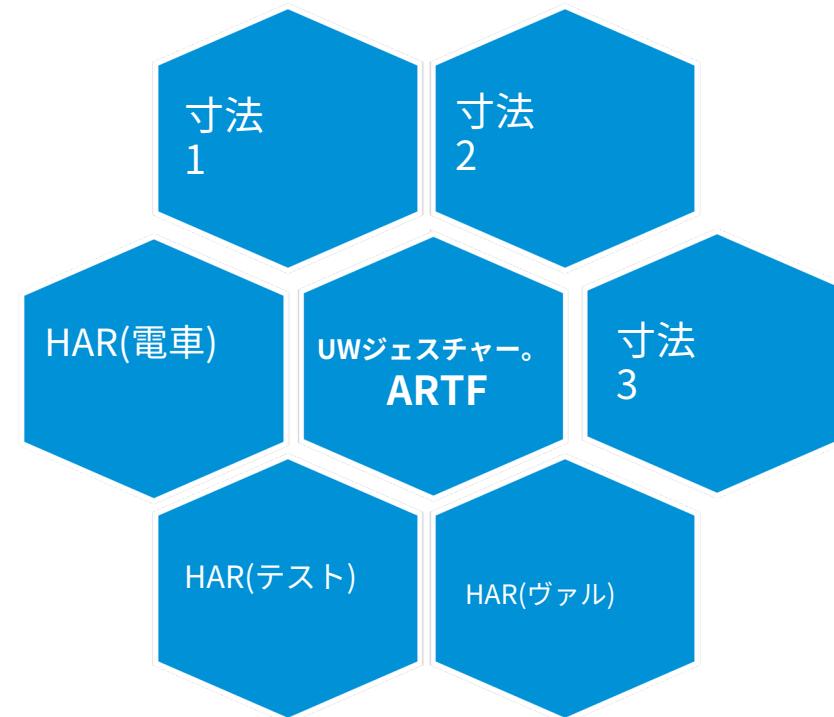


Fig. 4: Illustration of three common context-based methods: rotation, jigsaw, and colorization.



UWGesture ライブラリでトレーニングされ、HAR データ セットで微調整されたモデル

# タスク前処理

```
# Decode byte strings if needed
for column in df.select_dtypes([object]):
    df[column] = df[column].apply(lambda x: x.decode('utf-8') if isinstance(x, bytes) else x)

# Convert any numpy.ndarray values to lists
for column in df.columns:
    df[column] = df[column].apply(lambda x: x.tolist() if isinstance(x, np.ndarray) else x)

# Fill missing values using forward fill and backward fill
df = df.ffill().bfill()
```

```
def random_cropping(series, crop_length=None):
    """
    Crops a random sub-section of the time series.
    """
    if crop_length is None or crop_length > len(series):
        crop_length = len(series) // 2 # Default to half the length
    start = np.random.randint(0, len(series) - crop_length + 1)
    cropped_series = np.zeros_like(series)
    cropped_series[start:start+crop_length] = series[start:start+crop_length]
    return cropped_series
```

```
def time_warp(series, warp_factor=0.2):
    """
    Stretches or compresses the time series by a random factor within the specified range.
    """
    stretch_factor = 1 + (warp_factor * (np.random.rand() * 2 - 1)) # random factor between (1-warp_factor) and (1+warp_factor)
    warped_series = np.interp(np.arange(0, len(series)), stretch_factor, np.arange(0, len(series)), series)
    return warped_series[:len(series)] # Ensure same length after warping
```

```
def jitter(series, noise_level=0.05):
    """
    Adds small random noise to each point in the series.
    """
    noise = np.random.normal(0, noise_level, len(series))
    return series + noise
```

```
# Ensure numeric columns are of a correct type for scaling
numeric_columns = df.select_dtypes(include=['float64', 'int64']).columns
if len(numeric_columns) > 0:
    scaler = StandardScaler()
    df[numeric_columns] = scaler.fit_transform(df[numeric_columns])
```

## 変換とバックフィル

後方および前方のフィルテクニック - これは、中央値や平均値などは数値データと非数値データの両方に欠落値を埋める必要があるため、数値以外のデータに有効です。

時系列データセットでは欠損値が発生することがよくあるので、これを修正するために平滑化フィルタを適用します。ノイズを減らし、全体的な傾向を強調します。

## ランダムクロッピング

## タイムラップ

## ジッタリング

## データのスケーリングと正規化

時系列データの場合、系列の一部を引き伸ばしたり圧縮したりして、時間的な変化。

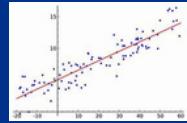
ジッタリングは、無関係な変動を無視し、コアパターンに焦点を当てることを学習することで、モデルの一般化を向上させるのに役立ちます。

SSLの口実タスクでは、元の時系列は、対照学習の正のペアとして使用でき、類似した時系列を認識するようにモデルをトレーニングできます。わずかなランダムノイズにもかかわらずパターン

正規化（最小最大スケーリング）：スケールデータを0から1の範囲に。

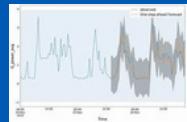
# 方法の選択

時系列に最適なモデルを見つけるために合計6つのモデルが使用されました  
データセット



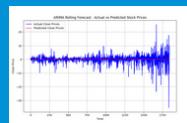
シムCLR (CL)

対照学習を使用して、同じ時系列の拡張ビュー間の一致を最大化します。



VIReg (MIL)

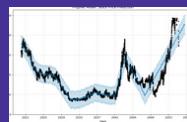
分散、不变性、共分散の損失を最小限に抑える多様かつ不变の表現を強制します。



生成的敵対

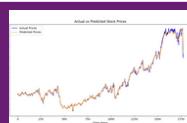
ネットワーク (GAN)

生成的敵対的ネットワーク (GAN) は、対照学習モデルが視点を学習できるように、画像の新しい変更されたビューを作成します。  
不变の特徴



TTTメソッド

テストデータがトレーニングデータセットと大きく異なる場合は、「組み合わせる」MAEを使用すると、欠落しているものを予測できるため、さらに効果的です。  
まずデータ内の情報です。"



CLIP (マルチモダリティ)

CLIPは、画像とテキストを整列させるように設計されたマルチモーダルモデルです。  
共有表現空間、オープンボキャブラリーイメージの実現  
認識と画像とテキストのマッチング。



マスク画像モデリング

MIMは視覚データを対象とした自己教師学習技術であり、「再構築」によって表現学習を改善することを目指している他の部分に基づいて画像の一部を作成します。

# マスク画像 モデリング

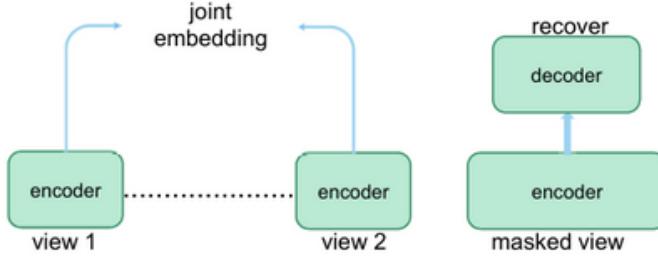
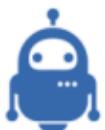


Fig. 7: The broad differences between CL and MIM. Note that the actual differences between their pipelines are not limited to what is shown.

## データの拡張、つまり分割 はなし

データの40~70%はマスクされ、残りはトレーニングされる  
それをエンコードすることで、  
データの特性



```
# Initialize masks and targets
mask = torch.zeros(batch_size, seq_length, dtype=torch.bool)
masked_data = data.clone()
target = data.clone()

for i in range(batch_size):
    # Randomly select indices to mask
    mask_indices = np.random.choice(seq_length, num_masked, replace=False)
    mask[i, mask_indices] = True
    masked_data[i, :, mask_indices] = 0 # Masking by setting the segment to zero
```

## 欠点：計算コストが高く、 70%のデータをランダムに選択してマスキングするデータ

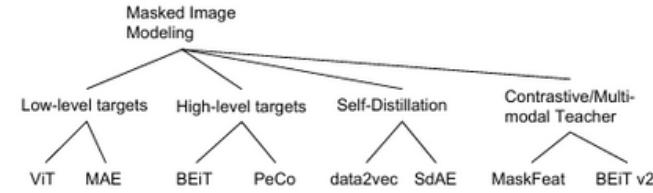
BEiT: NLP の BERT と同様に、BEiT は画像を小さな部分(トークン)に分割し、その一部を非表示にします。次に、視覚的なトークンを使用して、これらの欠落部分を予測しようとします。  
ピクセルターゲットトークンはモデルをトレーニングするためのターゲットトークンです。通常 25%程度の小さな画像部分で作業します。より大きなデータセットで使用されます。一般的に、複雑でより利用可能なデータに対して効率的かつ正確です。

### タスク

MAE - これは、画像の大部分(最大75%)を隠す、よりシンプルなモデルです。  
デコーダーを使用して、欠落している領域をピクセルから直接再構成します。画像を直接ピクセルとして使用します。画像のこの主要部分は 75%あります。一般的に小さなデータセットで使用されます。一般的に効率的で複雑性が低く、データタスクの可用性が低い場合に正確

### 2つの主要なALGO (BEiTとMAE)

BEiTは大規模なデータセットに使用され、MAEは小規模データセットに使用



VideoMAEとOmniMAEが使用される  
ビデオと同時モデル  
トレーニング。

### 分岐型MIMの図

MAEはイメージトークンを利用しません。代わりに、  
画像信号のスパース性の観点から問題に  
アプローチします。

# クリップ方式

欠点：時間的な関係を捉えられず、  
自然言語による記述に依存することが多い

マルチモーダル - テキスト、画像、ビデオなどのさまざま  
なデータソースを使用してトレーニングします。

モデル（時系列で使用）。プレミアム品  
質のタスクで使用されるため  
すべてのデータソースのノイズに対処しま  
す。技術には、フュージョン、クロスモーダル学  
習、注意メカニズムが含まれます。(CLIPの  
進歩によりマルチモーダル学習が大  
きく推進された)

## マルチモダリティ学習

マルチモーダル学習は乳児の学習メカニズ  
ムに自然に収束する

```
class CLIPModel(nn.Module):
    def __init__(self, ts_input_dim, ts_hidden_dim, ts_output_dim, text_input_dim, text_output_dim):
        super(CLIPModel, self).__init__()
        self.ts_encoder = TimeSeriesEncoder(ts_input_dim, ts_hidden_dim, ts_output_dim)
        self.text_encoder = TextEncoder(text_input_dim, text_output_dim)

    def forward(self, ts_data, text_data):
        ts_embedding = self.ts_encoder(ts_data)
        text_embedding = self.text_encoder(text_data)
        return ts_embedding, text_embedding

    def compute_loss(self, ts_embedding, text_embedding, temperature=0.07):
        # Normalize embeddings
        ts_embedding = F.normalize(ts_embedding, dim=1)
        text_embedding = F.normalize(text_embedding, dim=1)

        # Compute similarity matrix
        similarity_matrix = torch.matmul(ts_embedding, text_embedding.T) / temperature

        # Labels for contrastive loss
        batch_size = ts_embedding.size(0)
        labels = torch.arange(batch_size).to(ts_embedding.device)
```

CLIPは、CLスタイルの事前トレーニングタスクを利用して、キャプションと画像の対応を予測します。CLパラダイムの恩恵を受けて、CLIPはインターネットから収集された4億の画像とテキストのペアを含む大規模なデータセットでモデルをゼロからトレーニングすることができます。その結果、CLIPの進歩はマルチモーダル学習を研究の最前線に大きく押し上げました。



2024年現在、これについてはまだ研究が行われているが、これはあまり実行可能ではない。  
現時点では、これは時系列データの観点から最も正確性が高い。

# GAN (生成的 敵対的ネットワーク)

objective function [35], [143] is given as

$$\min_G \max_D V(G, D) = E_{x \sim p_{data}(x)} [\log D(x)] + E_{z \sim p_z(z)} [\log (1 - D(G(z)))] . \quad (18)$$

The SS-GAN [144] is defined by combining the objective functions of GANs with the concept of rotation [7]:

$$L_G(G, D) = -V(G, D) - \alpha \mathbb{E}_{x \sim p_G} \mathbb{E}_{r \sim R} [\log Q_D(R = r|x^r)], \quad (19)$$

$$L_D(G, D) = V(G, D) - \beta \mathbb{E}_{x \sim p_{data}} \mathbb{E}_{r \sim R} [\log Q_D(R = r|x^r)], \quad (20)$$

## SSL GANの目的関数 SS-GANと呼ばれる

ジェネレータ損失とディスクリミネータ損失が示されています。

ジェネレータは正のペアにできるだけ類似した画像を生成しようとし、ディスクリミネータは生成された画像をできるだけ区別しようとします。



対照的な方法は過剰につながる それと私んぐ 1つのん  
生成的手法はスケーラビリティにつながり、対照的生成と呼  
ばれる新しいパラダイムが考案されました。  
アルゴリズムとGANはその一部である

## 欠点: ハイパーパラメータとモデルに鈍感 崩壊

TABLE 3: Different losses of SSL.

Category	Method	Loss	Equation
Context-Based	Rotation [7]	Rotation Prediction	[3]
Pretext	MoCo v1 [50]	InfoNCE	[5]
CL	SimCLR v1 [52]	InfoNCE	[6]
	SimSiam [69]	Cosine Similarity	[9]
	Barlow Twins [55]	Invariance, and Covariance	[10]
	VICReg [56]	Variance, Invariance, and Covariance	[16]
Combinations with Other Learning Paradigms	SS-GAN [144]	GAN loss + Rotation Prediction	[19 & 20]
	S <sup>4</sup> L [145]	Supervised and Unsupervised Loss	[21]
	SSL improving robustness [146]	Supervised and Self-supervised Adversarial Training Loss	[22]
	unsupervised multi-view learning [64]	Self-supervised Loss on Multiple Views	[25]

## SSL のさまざまな損失表

```
# Generator Network
class Generator(nn.Module):
    def __init__(self, noise_dim, hidden_dim, output_dim, seq_length):
        super(Generator, self).__init__()
        self.fc1 = nn.Linear(noise_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, hidden_dim * seq_length // 2)
        self.fc3 = nn.Linear(hidden_dim * seq_length // 2, output_dim * seq_length)
        self.output_dim = output_dim
        self.seq_length = seq_length

    def forward(self, z):
        x = torch.relu(self.fc1(z))
        x = torch.relu(self.fc2(x))
        x = torch.tanh(self.fc3(x))
        return x.view(-1, self.output_dim, self.seq_length)

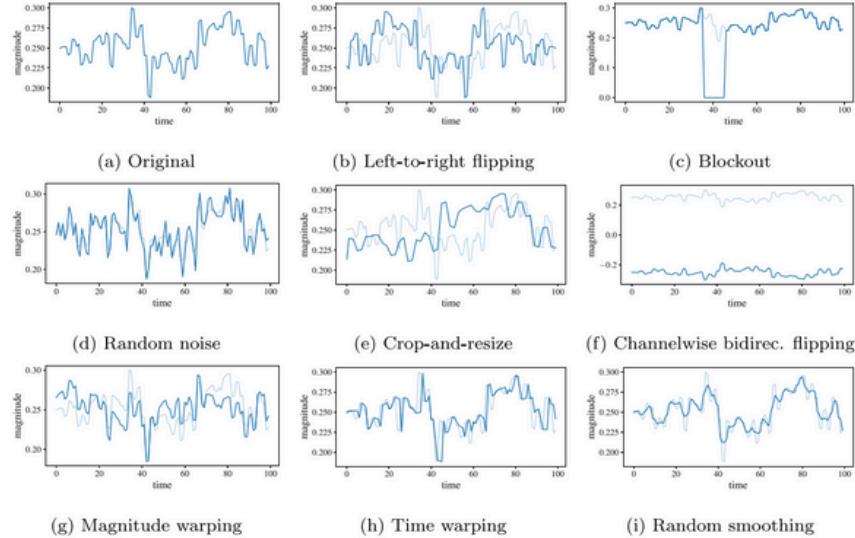
# Discriminator Network
class Discriminator(nn.Module):
    def __init__(self, input_dim, hidden_dim, seq_length):
        super(Discriminator, self).__init__()
        self.fc1 = nn.Linear(input_dim * seq_length, hidden_dim * seq_length // 2)
        self.fc2 = nn.Linear(hidden_dim * seq_length // 2, hidden_dim)
        self.fc3 = nn.Linear(hidden_dim, 1)

    def forward(self, x):
        x = x.view(x.size(0), -1) # Flatten time-series data
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return torch.sigmoid(self.fc3(x))
```

コードスニペット

# SimCLR(CL下)

欠点: 拡張感度が高く、コストが高い



## 増強方法

1. 左から右への反転: 時間シーケンスを逆にして、シーケンス順序に堅牢性を追加します。
2. ブロックアウト: 欠落したデータをシミュレートするために、系列の一部をランダムにゼロにします。
3. ランダムノイズ: 各チャンネルの分散に合わせてノイズを追加し、自然音に対する許容度を高めます。バリエーション。
4. 切り抜きとサイズ変更: ランダムにセグメントを取り取り、元の長さに補間して、異なるサブシーケンス。
5. マグニチュードワーピング: 正弦波パターンを使用して振幅を変化させ、自然な強度をシミュレートします変動。
6. タイムワーピング: シリーズの一部を引き伸ばしたり圧縮したりして、時間的な変化を導入します。
7. ランダムスマージング: スマージング フィルターを適用してノイズを減らし、全体的な傾向を強調します。

## プロセスの結果

モコブ2上に構築MoCo v1 および SimCLR v1、多層パーセプトロン (MLP) 投影ヘッドとより多くのデータを組み込んだ  
増強。

simCLRで行われる拡張は分類タスクには適していますが、物体認識タスクには適していません。そのため、強力な拡張は必須ですが、それだけに頼るのも良くありません。強力なデータ拡張によって生じる歪みは画像構造を変えてしまう可能性があります。  
その結果、弱く拡張された画像とは異なる分布となる。



```

# Left-to-right flipping
def left_to_right_flip(series):
    return np.flip(series, axis=1) # Flip across time axis (axis=1)

# Bidirectional flipping (mirroring across the time axis)
def bidirectional_flip(series):
    return -1 * series

```

```

# Blockout
def blockout(series, block_size_ratio=0.1):
    T = series.shape[1]
    block_size = int(block_size_ratio * T)
    start = np.random.randint(0, T - block_size)
    blocked_series = series.copy()
    blocked_series[:, start:start + block_size] = 0 # Block out along the time axis
    return blocked_series

```

```

# Random noise
def random_noise(series, noise_factor=0.1):
    sigma = np.std(series, axis=1, keepdims=True) # Compute std for each channel
    noise = np.random.uniform(-1, 1, series.shape) * sigma * noise_factor
    return series + noise

```

```

def magnitude_warping(series, magnitude=0.1):
    """Warp the magnitude of the time series"""
    series = np.array(series, dtype=np.float32) # Ensure it's a float32 array
    factor = 1 + np.random.uniform(-magnitude, magnitude)
    return series * factor

```

```

def time_warping(series, warp_factor=0.2):
    indices = np.arange(series.shape[1])
    warp_indices = np.random.uniform(0, warp_factor, series.shape[1])
    warped_series = np.interp(indices + warp_indices, indices, series)
    return warped_series

```

```

def random_smoothing(series, window_size=3):
    return np.convolve(series, np.ones(window_size)/window_size, mode='same')

```

$$\tilde{\mathbf{X}} = \mathbf{X}\boldsymbol{\gamma} \text{ where}$$

$$\boldsymbol{\gamma}_{i,j} = \begin{cases} 1, & \text{if } j = T - i + 1 \\ 0, & \text{if } j \neq T - i + 1 \end{cases}$$

$$\tilde{\mathbf{X}} = \boldsymbol{\Gamma}\mathbf{X} \text{ where}$$

$$\boldsymbol{\Gamma}_{i,j} = \begin{cases} 1, & \text{if } j = m - i + 1 \\ 0, & \text{if } j \neq m - i + 1 \end{cases}$$

For the blockout augmentation, we randomly select an area of  $\lambda \cdot m \in \mathbb{N}$  neighboring elements inside the time-series signal and set all values to zero. Starting in row  $k$  and column  $l$ , then

$$\tilde{x}_{i,j} = \begin{cases} 0 & \text{if } k \leq i \leq k + \lambda m \text{ and } l \leq j \leq l + \lambda m \\ x_{i,j} & \text{otherwise} \end{cases} \quad (8)$$

$$\tilde{\mathbf{X}} = \mathbf{X} + \mathbf{R} \circ \boldsymbol{\lambda} \boldsymbol{\Sigma},$$

$$\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_1, \dots, \boldsymbol{\Sigma}_m\} \in \mathbb{R}^{m \times T}$$

where  $\mathbf{X}$  denotes the original data and  $\tilde{\mathbf{X}}$  the augmented data,  $\boldsymbol{\Sigma}$  a column-vector composed of the per-channel standard deviations  $\sigma_i$  and  $\circ$  and element-wise multiplication.

;

$$\mathbf{X} = \mathbf{X} \circ \boldsymbol{\lambda} \boldsymbol{\Psi},$$

$$H(z) = \frac{\lambda}{2}z^1 + (1 - \lambda) + \frac{\lambda}{2}z^{-1}. \quad \boldsymbol{\Psi} = 1^{m \times T} + \sin \left( \begin{bmatrix} \frac{2\pi nt}{T} + \theta_1 \\ \vdots \\ \frac{2\pi nt}{T} + \theta_m \end{bmatrix} \right) \text{ where } t \in [0, \dots, T], \theta_i \in [0, \dots, 2\pi].$$

# VIReg (MIL)

## 欠点: 規制条件が多すぎて破綻しやすい

$Z^A$  and  $Z^B$ . The loss function of Barlow Twins is defined as

$$\mathcal{L}_{BT} = \sum_i (1 - C_{ii})^2 + \lambda \sum_i \sum_{j \neq i} C_{ij}^2. \quad (10)$$

Here,  $\lambda$  is a hyper-parameter, and  $C$  represents the cross-correlation matrix computed between the two batches of embeddings  $Z^A$  and  $Z^B$ , defined as

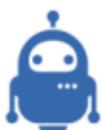
$$C_{ij} = \frac{\sum_b z_{b,i}^A z_{b,j}^B}{\sqrt{\sum_b (z_{b,i}^A)^2} \sqrt{\sum_b (z_{b,j}^B)^2}}, \quad (11)$$

where  $b$  indexes batch samples and  $i, j$  index the vector dimension of the networks' outputs.  $C$  is a square matrix that measures the correlation between the two batches of

### バーロウ・ツインズ

Barlow Twinsは、Reduce Redundancyアプローチを使用する自己教師学習 (SSL) 手法です。Barlow Twinsの目的は、埋め込みベクトルの各次元に固有の情報を持たせることです。

冗長性を削減します。



### トレーニングと微調整

- 各バッチごとに、2つの拡張ビューが生成されます。
- これらのビューの埋め込みは、VICReg 損失を介して渡されます。
- この損失を最小限に抑えるためにモデルのパラメータが更新されます。



VICRegは、分散正規化を導入することでBarlow Twinsアプローチを基盤としています。代表性の崩壊を防ぐ。3つの主要な目的のバランスをとる。

1. **分散:** 埋め込みの各次元の標準偏差が閾値 $\gamma$ 以上になるようにし、低い分散を維持するようにペナルティを課す。  
多様性。
2. **不变性:** 埋め込み間の平均二乗ユークリッド距離を使用して、拡張ビュー間の類似性を強化し、類似したビューの距離を最小化します。  
サンプル。
3. **共分散:** 埋め込みの共分散行列を計算し、非対角要素（次元間の相関）にペナルティを課し、独立した  
特徴。

```
# Fine-tuning loop
def fine_tune(model, dataloader, criterion, optimizer, epochs=10):
    model.train()

    for epoch in range(epochs):
        epoch_loss = 0.0
        correct_preds = 0
        total_preds = 0

        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)

            # Forward pass
            outputs = model(inputs)
            loss = criterion(outputs, labels)

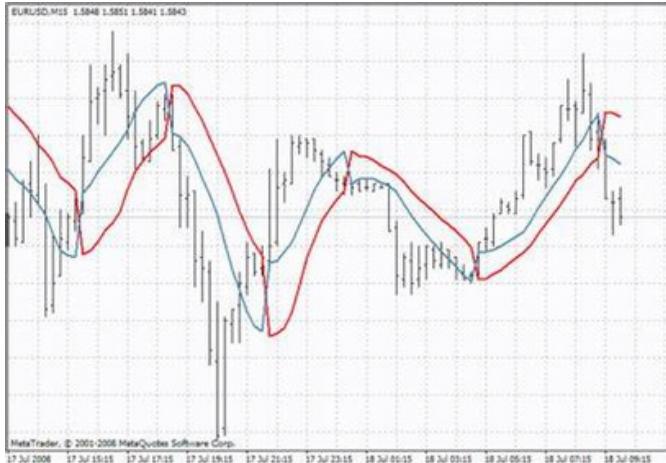
            # Backpropagation
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

            # Track loss and accuracy
            epoch_loss += loss.item()
            _, predicted = torch.max(outputs, 1)
            correct_preds += (predicted == labels).sum().item()
            total_preds += labels.size(0)

        accuracy = 100 * correct_preds / total_preds
        print(f"Epoch [{epoch + 1}/{epochs}], Loss: {epoch_loss / len(dataloader):.4f}, Accuracy: {accuracy:.2f}
```

# 評価指標

すべての方法に長所がありますが、SimCLR と ViReg は複雑な関係と時間的ダイナミクスを効果的にモデル化できる点で際立っています。



## 指標

下流タスクのパフォーマンス

k-NN分類

ネットワークの分析

## 説明

下流タスク評価：モデルを動物の分類タスク（猫、犬、鳥や鳥など）うまく機能すれば、学習したことがわかります便利な機能。

K-NN分類の精度は、学習した特徴空間がどれだけよく分離されているかを示すことができ、トレーニングする追加の分類器。

ネットワークを調べると、モデルの特定の部分が動物の毛皮や羽毛、さらには斑点や縞模様のような特定のパターン。これは、SSLがモデルに動物特有の特徴を理解するのに役立ったことを示しています。これには、モデルが何を学習したかを確認するために各ニューロンを分解することが含まれます。

これらの指標は、異なる環境間でうまく機能する有意義で移転可能な特徴を学習するSSLモデルの有効性を評価するのに役立ちます。

トレーニング中にラベル付けされたデータが使用されなかった場合でも、タスクは実行されます。

Contrastive Lossはコードスニペットで使用されています

## 相関分析

相関係数を使用して相関の高い特徴を特定し、  
保持のみ

遅れたペア。

### 期待される結果:

予測精度の向上: 特徴量を適用した後、モ  
デルは精度指標 (MAE、MSE、RMSEの低  
下など) が向上することが期待されます。

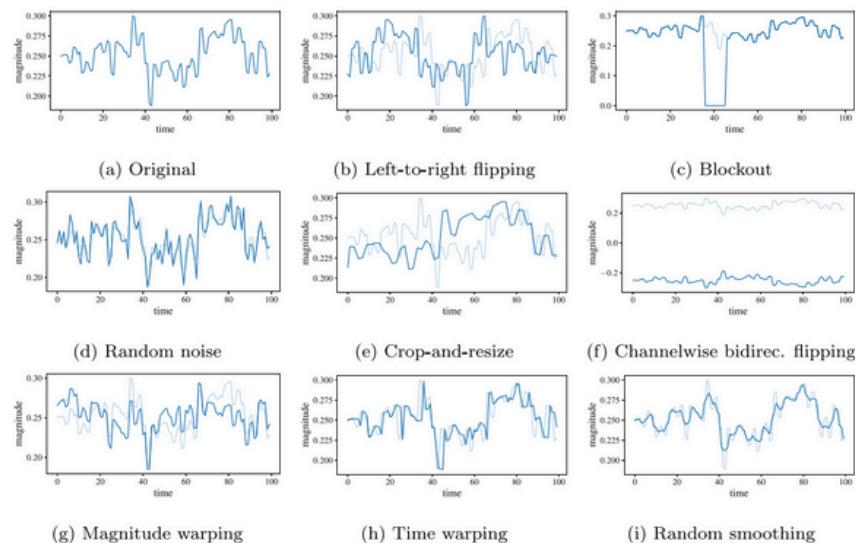
選択。

# 機能の重要性

微調整では、特徴と重みの重要度は、次の点に焦点を当てることで、事前学習済みモデルを新しいタスクに適応させるのに役立ちます。

関連するパターンを抽出し、それに応じて重みを調整します。特徴の重要度により、モデルは時系列分類の傾向など、新しいデータに固有のパターンを優先できます。一方、重みの重要度により、モデルのパラメータを選択的に調整できます。下位層は事前トレーニングからの一般的なパターンを保持することが多く、変更は最小限で済みますが、上位層はタスク固有のニュアンスに適応します。このプロセスにより、

最初から完全に再トレーニングすることなく、新しいデータセットでのモデルのパフォーマンスをテストします。



## 期待される結果:

トレーニング時間の短縮：機能セットが削減されたため、トレーニングプロセスが速くなり、より多くの効率的な実験。

# 欠損値の処理

不正確な予測や偏った結果につながる可能性があります。適切なモデル、または KNN 補完などの高度な手法を使用すると、貴重な情報を得ることができます。

## 期待される結果:

欠損値を適切に処理したデータセットでトレーニングされたモデルは、

未解決の行方不明者への  
価値観。

## 期待される結果:

外れ値が除去されたデータセットでトレーニングされたモデルは、エラー率が低くなり、予測パフォーマンスが向上します。

# 外れ値の除去

外れ値の影響: 外れ値は結果を歪め、モデルの学習プロセスを誤らせる可能性があります。Zスコアや IQRなどの技術を使用して外れ値を特定して除去すると、データを安定させることができます。  
分布。

# 正規化とスケーリング

特徴の正規化または標準化（例：最小値、すべての特徴がモデルtに等しく寄与する

保証する  
veに

特徴



## 期待される結果:

- 正規化されたデータセットを使用するモデルは、より良い収束を示し、未加工のスケーリングされていない機能を使用した場合と比較したパフォーマンス メトリック。



## 期待される結果:

厳選された機能のサブセットは、より高い利用可能なすべてのモデルを使用したモデルと比較して、精度と複雑さが低い特徴。

# まとめ

このプロジェクトの目的は、UWaveGestureLibraryデータセットを自己教師あり学習 (SSL) を使用して作成します。SSLは大規模なラベル付けされていないデータから有用な特徴表現を学習し、下流の分類を改善できるパフォーマンス。このプロジェクトでは、自己教師あり学習法のVICRegとSimCLRを使用して別のデータセットで事前学習を行いました。VICRegは、分散、不变性、共分散の正規化により、堅牢で多様な特徴抽出を保証します。一方、対照学習アプローチのSimCLRは、同じインスタンスの拡張ビューに対して同様の表現を促します。事前学習後、UWaveGestureLibraryデータでモデルを微調整して分類精度を最大化し、学習した特徴をジェスチャー認識タスクに効果的に適応させました。このアプローチは、モデルの精度を向上させるのに役立ちます。

重要な時系列特性を捉える SSL 抽出表現を使用します。

# 前処理技術と外れ値の除去

これらの技術は、モデルの予測精度を高めるだけでなく、より信頼性が高く堅牢な予測を保証します。

データ内のノイズや無関係な変動を最小限に抑えることによって、次のことが将来の見通しにどのように影響するかを説明します。

```
# 1. **Smoothing using Rolling Window (moving average)**
df['Close_smoothed'] = df['Close'].rolling(window=5).mean()

# 2. **Outlier Removal using Z-Score**
z_scores = np.abs(stats.zscore(df['Close_smoothed'].fillna(df['Close'])))
df = df[(z_scores < 3)] # Removing outliers where z-score > 3

# 3. **Scaling (MinMaxScaler) for features**
scaler = MinMaxScaler()
df[features] = scaler.fit_transform(df[features])
df[[target]] = scaler.fit_transform(df[[target]])
```

- 平滑化手法（移動平均、指数平滑化など）は、短期的な変動を取り除くのに役立つ。  
変動と長期的な傾向を強調する  
時系列データ
- 今後の展望: より洗練されたスムージング  
季節分解やウェーブレット変換などの技術を使用して、時系列を傾向、季節、および  
残留成分を摂取する前に、  
トレンド予測の精度を向上します。

# 今後の展望

GANとCLIPは間違いなく未来の展望となる

Methods	Linear Probe	Fine-Tuning	VOC_det	VOC_seg	COCO_det	COCO_seg	ADE20K_seg	DB
Random:	17.1 <sub>A</sub> [18]	-	60.2 <sub>R</sub> [69]	19.8 <sub>A</sub> [18]	36.7 <sub>R</sub> [50]	33.7 <sub>R</sub> [50]	-	-
R50 Sup	76.5 [68]	76.5 [68]	81.3 <sup>e</sup> [69]	74.4 [67]	40.6 [50]	36.8 [50]	-	-
ViT-B Sup	82.3 [70]	82.3 [70]	-	-	47.9 [70]	42.9 [70]	47.4 [70]	-
<b>Context-Based:</b>								
Jigsaw [8]	45.7 <sub>R</sub> [68]	54.7	61.4 <sub>R</sub> [42]	37.6	-	-	-	256
Colorization [38]	39.6 <sub>R</sub> [68]	40.7 [7]	46.9	35.6	-	-	-	-
Rotation [7]	38.7	50.0	54.4	39.1	-	-	-	128
<b>CL Based on Negative Examples:</b>								
Exemplar [132]	31.5 [48]	-	-	-	-	-	-	-
Instdisc [48]	54.0	-	65.4	-	-	-	-	256
MoCo v1 [50]	60.6	-	74.9	-	40.8	36.9	-	256
SimCLR [52]	73.9 <sub>V</sub> [82]	-	81.8 <sup>e</sup> [69]	-	37.9 [69]	33.3 [69]	-	4096
MoCo v2 [51]	72.2 [69]	-	82.5 <sup>e</sup>	-	39.8 [56]	36.1 [56]	-	256
MoCo v3 [82]	76.7	83.2	-	-	47.9 [70]	42.7 [70]	47.3 [70]	4096
<b>CL Based on Clustering:</b>								
SwAV [68]	75.3	-	82.6 <sup>e</sup> [56]	-	41.6	37.8 [56]	-	4096
<b>CL Based on Self-distillation:</b>								
BYOL [67]	74.3	-	81.4 <sup>e</sup> [69]	76.3	40.4 [56]	37.0 [56]	-	4096
SimSiam [69]	71.3	-	82.4 <sup>e</sup> [69]	-	39.2	34.4	-	512
DINO [83]	78.2	83.6 [98]	-	-	46.8 [100]	41.5 [100]	44.1 [99]	1024
<b>CL Based on Feature Decorrelation:</b>								
Barlow Twins [55]	73.2	-	82.6 <sup>e</sup> [56]	-	39.2	34.3	-	2048
VICReg [56]	73.2	-	82.4 <sup>e</sup>	-	39.4	36.4	-	2048
<b>Masked Image Modeling (ViT-B by default):</b>								
Context Encoder [104]	21.0 <sub>A</sub> [7]	-	44.5 <sub>A</sub> [7]	30.0 <sub>A</sub>	-	-	-	-
BEiT v1 [99]	56.7 [111]	83.4 [98]	-	-	49.8 [70]	44.4 [70]	47.1 [70]	2000
MAE [70]	67.8	83.6	-	-	50.3	44.9	48.1	4096
SimMIM [101]	56.7	83.8	-	-	52.3 <sub>Swin-B</sub> [244]	-	52.8 <sub>Swin-B</sub> [244]	2048
PeCo [107]	-	84.5	-	-	43.9	39.8	46.7	2048
iBOT [98]	79.5	84.0	-	-	51.2	44.2	50.0	1024
MimCo [110]	-	83.9	-	-	44.9	40.7	48.91	2048
CAE [100]	70.4	83.9	-	-	50	44	50.2	2048
data2vec [108]	-	84.2	-	-	-	-	-	2048
SdAE [109]	64.9	84.1	-	-	48.9	43.0	48.6	768
BEiT v2 [111]	80.1	85.5	-	-	-	-	53.1	2048

自己教師ありGAN (SS-GAN) アプローチを活用することで、タスクも使用できるようになります。

特定の変換（時間シフトや回転など）をさらに

モデルの特徴学習を強化する。マルチモーダル設定でCLIP (Contrastive Language-Image Pretraining) を統合することで、テキストデータやカテゴリデータが時系列データと組み合わされている場合、

モデルの文脈理解を強化する。CLIPは視覚言語タスクによく使用されるが、その原理を時系列信号に適応させることで、

テキストによる説明（イベントラベルやメタデータなど）を追加すると、

時間依存のイベントの解釈可能性と分類精度。GAN生成データとCLIPにヒントを得たマルチモーダル学習の組み合わせにより、さまざまな時系列にわたってモデルを一般化する能力が大幅に向かう可能性がある。

分類タスク。

VICRegとSimCLRを使用して時系列データをトレーニングおよび微調整した場合の2つの潜在的な仮説を以下に示します。

データ：

仮説1：VICRegの冗長性削減は時系列データの特徴の多様性を高めるVICRegの分散、不变性、共分散への焦点を活用することで、モデルはより堅牢な多様な特徴セットを生み出し、

時系列データのさまざまな側面を分析することで、微調整によって分類精度が向上するはずです。

UWaveGestureLibrary データセットは、機能表現における冗長性と崩壊を回避するのに役立ちます。

仮説2：SimCLRの拡張と対照学習はジェスチャークラスの区別を改善する強力なデータ拡張に依存するSimCLRの対照学習アプリーチは、モデルが

ジェスチャークラスをより効果的に区別します。ジッタリングや時間ベースのシフトなどの拡張機能を通じて、SimCLRはモデルが不变表現を学習し、類似しているが異なる時間分類能力を向上させるのに役立ちます。

シリーズパターン。