Analysis of Causal and Non-Causal Convolution Networks for Time Series Classification

Uday Singh Saini* usain001@ucr.edu
Zhongfang Zhuang† zzhuang@visa.com
Chin-Chia Michael Yeh† miyeh@visa.com
Wei Zhang† wzhan@visa.com
Evangelos E. Papalexakis* epapalex@cs.ucr.edu

Abstract

Applications of neural networks like MLPs and ResNets in temporal data mining has led to improvements on the problem of time series classification. Recently, a new class of networks called Temporal Convolution Networks (TCNs) have been proposed for various time series tasks. Instead of time invariant convolutions they use temporally causal convolutions, this makes them more constrained than ResNets but surprisingly good at generalization. This raises an important question: How does a network with causal convolution solve these tasks when compared to a network with acausal convolutions? As the first attempt at answering these questions, we analyze different architectures through a lens of representational subspace similarity. We demonstrate that the evolution of input representations in the layers of TCNs is markedly different from ResNets and MLPs. We find that acausal networks are prone to form groupings of similar layers and TCNs on the other hand learn representations that are much more diverse throughout the network. Next, we study the convergence properties of internal layers across different architecture families and discover that the behaviour of layers inside Acausal network is more homogeneous when compared to TCNs. Our extensive empirical studies offer new insights into internal mechanisms of convolution networks in the domain of time series analysis and may assist practitioners gaining deeper understanding of each network.

Keywords: time series classification, model representation analysis, subspace clustering, transparent AI.

1 Introduction

The Time Series Analysis domain [28, 9, 1] is a growing and challenging arena of machine learning. With applications and data sources originating from diverse domains such as healthcare, cyber-security, weather forecasting and climate analysis, tackling the time series classification problem can potentially impact in many areas of engineering and science [6]. Traditionally, this problem has been tackled by a wide array of Non Deep-Learning algorithms [14, 16, 4, 26, 2].Recently, multilayer perceptrons (MLP) [28, 10, 12] and ResNets [13, 28, 10] have been adapted for the problem of time series classification[28]. Both, MLP and ResNets[28] are agnostic to the temporal nature of the underlying data. To circumvent these issues, [3] propose a causal convolution operation, where each convolution filter operates on a given time step and its antecedents. Using this setup, they capture long range dependencies in a sequence and demonstrate competitive behaviour on sequential datasets, their approach is also referred to as Temporal Convolution Networks or TCNs.

In this study we utilize tools from subspace clustering literature[17] and conducted an investigative analysis into the hidden representations of neural networks along the lines of [15, 25, 23]. Such an analysis of feed-forward neural networks is absent in the domain of Time Series Classification and could aid in the development of architectures tailored for this domain.

The chief goal of this study is to analyze how a TCN differs from a ResNet or an MLP, both of with employ Acausal Operations as opposed to a TCN. Do Acausal Convolutions learn similar inductive priors to Causal Convolutions? If not, then how do Acausal and Causal Convolutions affect the internal behaviour of their respective networks? Via the medium of this study we explore such issues and motivate Least Squares Regression Subspace Clustering (LSRSC)[17] as a tool to analyze the representations inside the layers of a network. We then combine LSRSC with Centered Kernel Alignment (CKA)[5] along the lines of Linear-CKA[15, 19, 23, 21] and SSC-CKA [25] and in Section 4 we offer comparisons between LSRSC-CKA and related works like Linear-CKA and SSC-CKA.Our main contributions are as follows:

- Our investigations reveal major differences between

*University of California, Riverside
†Visa Research

the internal representations of MLPs, ResNets and TCNs. More specifically we observe that the less restrictive the inductive prior of an architecture, the more prominent the *block-structure*[15, 19, 21] in the network's internal representation structure.
- Upon performing a direct comparison of TCNs with ResNets we observe that only the initial layers of the 2 architectures share any similarities, and their respective representations quickly diverge as a function of depth. This phenomena is not so extreme when comparing networks within the same architecture family.
- Finally we demonstrate that the rate at which different layers converge to their final representations across different architectures is distinct. More specifically we observe that while for Acausal architectures the shallower layers are among the first to converge and the final layers among the last, layers of TCNs are generally a bit more dynamic in their convergence orders.

2 Background and Experimental Goals

The goal in this study is to analyze the behaviour of neural network architectures with different inductive priors for the task of time series classification. To this effect, parallel to the lines of SSC-CKA[25] we motivate an amalgamation of LSRSC[17] with CKA[5], henceforth referred to as LSRSC-CKA. Next, we lay down preliminary background and outline the algorithmic setup used to analyze neural network architectures.

2.1 Least Squares Regression Subspace Clustering: Given a Matrix $X = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$ where $X \in \mathbf{R}^{d \times N}$, the goal of LSRSC[17] is to learn a set of Affinity Coefficients, denoted by $C \in \mathbf{R}^{N \times N}$. The aim here is to help reconstruct each data point as a Linear Combination of other data points.The authors of LSRSC[17] motivate the following LSR problem accounting for Noise as described in Equation 2.1, which they then solve via the steps described in Equation 2.2. A good solution to Equation 2.1 will focus on reducing the relative error as defined in Equation 2.3.

$$\min_{C} ||X - XC||_F^2 + \lambda ||C||_F^2 \quad \text{s.t.} \quad diag(C) = 0 \tag{2.1}$$

$$\begin{aligned} C^* &= -D(diag(D))^{-1} \text{s.t.} \quad diag(C^*) = 0 \\ D &= (X^T X + \lambda I)^{-1} \end{aligned} \tag{2.2}$$

$$\text{Relative Error} = \frac{||X - XC||_F}{||X||_F} \tag{2.3}$$

2.2 Centered Kernel Alignment: Given 2 pairwise similarity matrices over N examples, namely, $C_1 \in \mathbf{R}^{N \times N}$ and $C_2 \in \mathbf{R}^{N \times N}$, Centered Kernel Alignment[5, 15, 20, 23, 21] as defined in Equation 2.4 is an isotropic invariant similarity metric over 2 matrices.

$$\text{CKA}(C_1, C_2) = \frac{\text{HSIC}(C_1, C_2)}{\sqrt{\text{HSIC}(C_1, C_1)\text{HSIC}(C_2, C_2)}} \tag{2.4}$$

Where Hilbert-Schmidt Independence Criterion (HSIC)[11] is a statistical test to determine the independence of 2 sets of variables, in this case C_1 and C_2 from Equation 2.4. HSIC measures the similarity of 2 matrices by computing a scaled dot product between the vectorized representations of centered similarity matrices as shown in Equation 2.5.

$$\text{HSIC}(C_1, C_2) = \frac{\text{vec}(HC_1H) \cdot \text{vec}(HC_2H)}{(N-1)^2} \tag{2.5}$$

Here $H = I - \frac{1}{N}\mathbf{1}\mathbf{1}^T$ is a centering matrix.

Algorithm 1: Layer-wise LSRSC Computation

Data: Layer-wise Neural Activation Matrices: $[X_1, \ldots, X_l]$
Result: List of Layer-wise Affinity Matrices: $\mathbf{C} = [C_{X_1}, \ldots, C_{X_l}]$

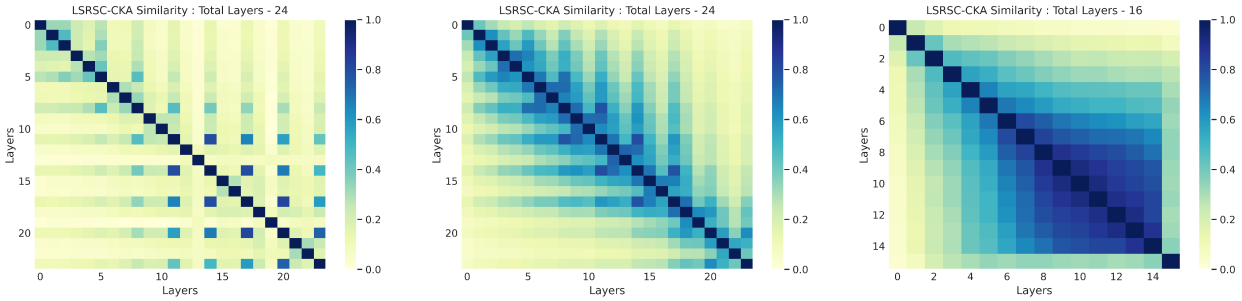
```
initialization: C = [];
for i ← 1 to l do
    For X_i, Compute C_i based on Equation 2.2;
    Compute - C_{X_i} = |C_i| + |C_i|^T;
    C.append(C_{X_i});
end
```

Algorithm 2: All-pairs CKA Computation

Data: Layer-wise Affinity Matrices: $\mathbf{C} = [C_{X_1}, \ldots, C_{X_l}]$
Result: All-pairs similarity: $S \in \mathbf{R}^{l \times l}$.

```
initialization: S = 0;
for i, j in permutations(l, 2) do
    S_ij = CKA(C_{X_i}, C_{X_j}) - Equation 2.4;
end
```

2.3 Algorithmic Pipeline: Now we describe how we combine LSRSC and CKA and utilize it for the task of analyzing neural network architectures. Let $X_i \in \mathbf{R}^{d_i \times N}$ and $X_j \in \mathbf{R}^{d_j \times N}$ be 2 matrices that represent the latent representations of N examples given by any 2 layers of a Neural Network, namely - layer i and j. Given a layer's activations we then proceed to compute the corresponding pairwise subspace affinity



(a) LSRSC-CKA: TCN - 24 Layers. Accuracy - 95.45 %. Inductive Prior - Causal Convolution

(b) LSRSC-CKA: ResNet - 24 Layers. Accuracy - 93.56 %. Inductive Prior - Acausal Convolution

(c) LSRSC-CKA: MLP - 15 Layer. Accuracy - 74.54 %. Inductive Prior - Fully Connected

Figure 1: LSRSC-CKA Heatmap. Networks from Left to Right : TCN 24-Layer, ResNet 24-Layer and MLP 15-Layer. Dataset - FordA, Dataset size - 1320 Examples x 500 Time steps per example. Inductive Priors from Left to Right - Causal Convolution (TCN), Acausal Convolution (ResNet) and Fully Connected (MLP).

matrix C_{Xi} for that X_i based on Equation 2.2 as described in LSRSC[17] and summarized in algorithm 1. Having obtained a list of layer-wise affinity matrices C_{Xi} we then proceed to compute similarities between all pairs, denoted by S, of a network's l layers, as outlined in algorithm 2. The results of this process to learn a layer-wise similarity matrix for a network can be seen for different networks, for instance in Figure 1 of Section 3 for TCN, ResNet and MLP respectively.

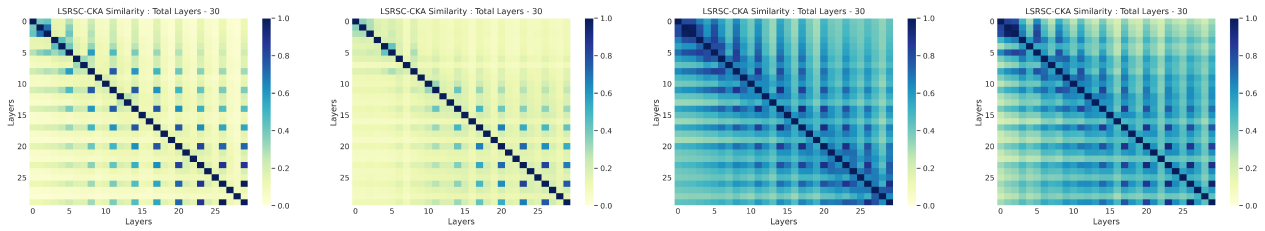
3 Experiments for network inductive prior analysis

Temporal Convolution Networks (TCNs) utilize a fundamentally different class of operations than ResNets and MLPs and yet achieve similar performance to these networks [3]. In this section we use LSRSC-CKA to analyze the behaviour of Temporal Convolution Networks by comparing them with other acausal networks used for time series classification. We demonstrate that the presence of a task appropriate inductive prior in the form of causal convolution helps TCNs avoid common architectural pathologies like the aforementioned Block-Structure often associated with overparameterized neural networks[15, 19, 25]. We compare TCNs with ResNets and demonstrate that networks with the different inductive priors behave differently and vice versa. Further, we demonstrate that Internal layers of TCNs behave differently from Acausal Networks in terms of their learning trajectories, which is a consequence of how information propagates within these architectures.

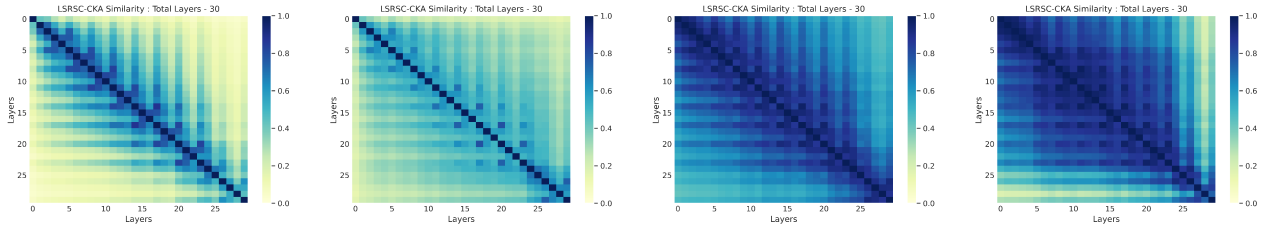
3.1 Analyzing the effects of various Inductive Priors To begin our investigation on the effects of various inductive priors on a network's internal representation structure we train different networks on the FordA dataset from UCR Time Series Repository[6]. We train TCNs, ResNets and MLP as shown in Figure 1a - Figure 1c, respectively. We find that Acausal networks like ResNets and MLPs tend to have a higher prevalence of Block-Structure endemic to overparameterized models [15, 19, 25] in their pairwise layer similarity map.In particular, the deeper layers of an MLP start becoming similar to each other and additional layers don't learn new representations. We further reinforce these findings in Section 3.3, where we show the same behaviour with increasing network depth. This problem is alleviated to an extent in ResNets where a deeper layer is increasingly dissimilar to a shallower layer, thereby indicating that each new layer is building upon the representations of the previous layer.

Focusing on TCNs in Figure 1a we observe a sparse structure towards the deeper layers where those additional layers tend to learn representations that are unique, this is courtesy of dilated temporal convolution used in TCNs where each such convolution filter looks at a unique causal sub-sequence of the input. This operation is markedly different from the other 2 where each convolutional filter of a ResNet is locally invariant in its causality and the non-linear transform of the MLP assumes all possible interactions in the entire sequence, which makes it particularly prone to overfitting. The performance gains as a result of different inductive priors are also self evident as the TCN and ResNet have accuracies of 95.45% and 93.56% respectively, while the MLP saturates at 74.54%.

3.2 Examining the effects of an Inductive Prior over different datasets Next, to consolidate our findings in Section 3.1 on the role of various inductive priors.



(a) LSRSC-CKA: TCN - 30 Layers. Accuracy - 83.09 %. Dataset - FordB **(b)** LSRSC-CKA: TCN - 30 Layers. Accuracy - 56.21 %. Dataset - IWB **(c)** LSRSC-CKA: TCN - 30 Layers. Accuracy - 83.47 %. Dataset - LKA **(d)** LSRSC-CKA: TCN - 30 Layers. Accuracy - 73.86 %. Dataset - SKA



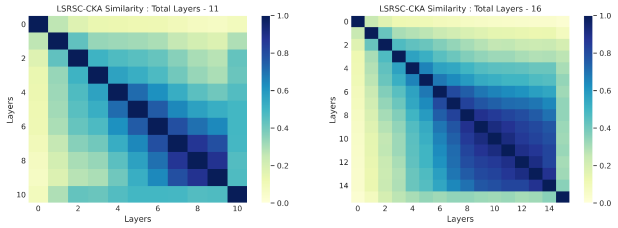
(e) LSRSC-CKA: ResNet - 30 Layers. Accuracy - 82.35 %. Dataset - FordB **(f)** LSRSC-CKA: ResNet - 30 Layers. Accuracy - 53.38 %. Dataset - IWB **(g)** LSRSC-CKA: ResNet - 30 Layers. Accuracy - 88.80 %. Dataset - LKA **(h)** LSRSC-CKA: ResNet - 30 Layers. Accuracy - 80.26 %. Dataset - SKA

Figure 2: LSRSC-CKA Heatmap, Networks : Row 1 - TCN and Row 2 - ResNets. Datasets (From Left to Right in each row) - FordB, IWB, LKA . Evaluation Dataset Sizes : FordB Size - 810 Examples x 500 Time Steps per example, Insect Wing Beats Sound or IWB - 1980 Examples x 256 Time steps per example, Large Kitchen Appliances or LKA - 375 Examples x 720 Time steps per example.

We perform an additional set of experiments, this time on datasets like FordB - Size : 405,000 Cumulative Time Steps, InsectWingbeatSound - Size : 506,880 Cumulative Time Steps, and Large Kitchen Appliances - Size : 270,000 Cumulative Time Steps, from the UCR Time Series Repository[6] as shown in Figure 2. In doing so, we reaffirm the same phenomena, i.e. a much sparser block structure for TCNs as shown in Figure 2a - Figure 2c and the Fish-Tailed Block Structure for ResNets Figure 2e - Figure 2g. We omit MLPs due to their lack of competitive results for reasons described earlier. These recurrences highlight the consistency with which different network architecture families will behave due to factors inherent to their architectures . In Figure 2c and Figure 2g we also observe that the same architecture for smaller datasets also contains a background block structure just by the virtue of having more parameters relative to the data, a phenomena we which also touch upon later in Section 4.

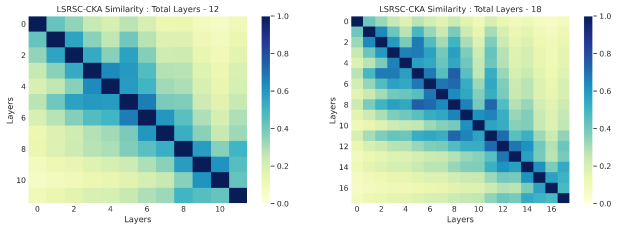
3.3 Effects of Network depth over different inductive priors To consolidate our findings in Section 3.1 next we evaluate and demonstrate the behaviour of each network family with varying network depth. Given an architecture, scaling network depth has been shown to cause deeper layers of a network to be very similar to each other [15, 19, 25, 21, 23]. This phenomena is also called the *Block-Structure* and a sign of network over-parameterization with respect to the data[19] and can be a sign of a few data points having large representations in the activation space of those layers[21]. In our analysis we find that MLPs and ResNets, Figure 3 and Figure 4 respectively, evolve some form of a dominant *Block-Structure* when compared to TCNs in Figure 5. The experiments shown were conducted on the FordA dataset[6]. To further elaborate, we begin by analyzing 10 and 15 layer Multi-layer Perceptrons. We observe that as we go deeper the block diagonal structure becomes more prominent and the final few internal layers of the network become increasingly similar to each other. This shows that adding extra learning capacity in a network doesn't necessarily the network learn newer representations and translate into improved performance as shown in Figure 3. Similar observations were also made for Image Classification in [15, 25]. Progressing the analysis to ResNets in Figure 4 we observe a block-structure which is distinct from MLPs but still indicates long range similarities between a layer and its ancestors. Though MLPs and ResNets are both acausal networks in nature, ResNets, having fewer parameters than MLPs don't tend to suffer from poor generalization that MLPs do. This is indicated both in consistent generalization performance and similar layerwise similarity patterns across network depths

in Figure 4. In stark contrast to MLPs and ResNets, the architecture of TCNs learns representations that are much more distinct throughout the depth of the network and also achieve good generalization performance, as shown in Figure 5.



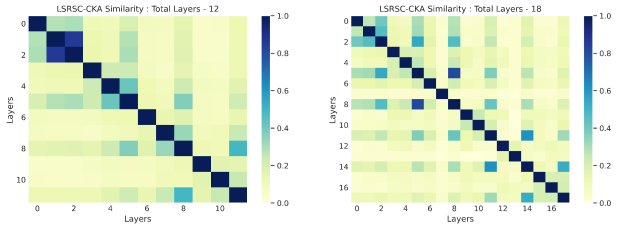
(a) LSRSC-CKA: MLP - 10 Layer. Accuracy - 76.28 % **(b)** LSRSC-CKA: MLP - 15 Layer. Accuracy - 74.54 %

Figure 3: LSRSC-CKA Heatmap, Networks - MLPs with width - 50 , Dataset - FordA, Evaluation Dataset size - 1320 Examples x 500 Time steps per example, Inductive Prior - Fully Connected.



(a) LSRSC-CKA: ResNet - 12 Layers. Accuracy - 93.83 % **(b)** LSRSC-CKA: ResNet - 18 Layers. Accuracy - 93.36 %

Figure 4: LSRSC-CKA Heatmap, Networks - ResNets , Dataset - FordA, Evaluation Dataset size - 1320 Examples x 500 Time steps per example, Inductive Prior - Acausal Convolution

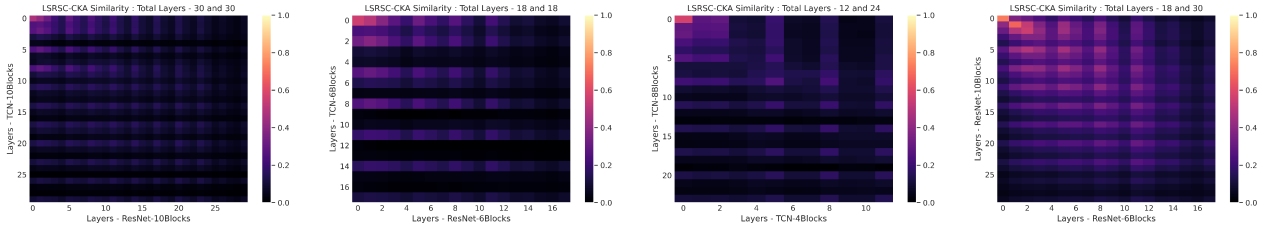


(a) LSRSC-CKA: TCN - 12 Layers. Accuracy - 95.15 % **(b)** LSRSC-CKA: TCN - 18 Layers. Accuracy - 95.45 %

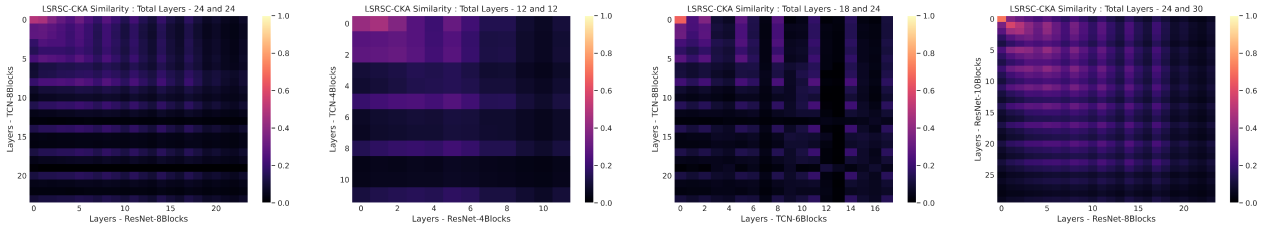
Figure 5: LSRSC-CKA Heatmap, Networks - TCN , Dataset - FordA, Evaluation Dataset size - 1320 Examples x 500 Time steps per example , Inductive Prior - Causal Convolution

3.4 Comparing different neural networks architectures In algorithm 2 we described the procedure to compare the layers of the same network with each other. However LSRSC-CKA just like Linear-CKA can also compare any 2 layers coming from different architectures. This is because, as described in Section 2, LSRSC helps us distill any Activation matrix over N examples into an affinity matrix over those examples, thus enabling comparisons between 2 affinity matrices over the same set of inputs. Earlier in Section 3.1 we compared TCNs and ResNets as 2 separate entities. In this section, along the lines of Linear-CKA[24, 21] we utilize LSRSC-CKA's ability to compare 2 different architectures directly and present the results in Figure 6 for FordA dataset. In Figure 6a and Figure 6e, We first perform this comparison between a TCN and a ResNet, which is the central subject of focus in our study. On the vertical axis are the layers of a TCN and the horizontal axis is spanned by the layers of a ResNet. We observe that only the very earliest layers of both the networks share any significant similarity by the virtue of being close to the input, and as depth increases their representations start to diverge. This gradual diversion of similarity points to the fact that as the layers of the respective networks get deeper, the subspaces learnt to represent input instances become different. This augments our findings in Section 3.1 where in Figure 1 we first demonstrated architectural differences between TCNs and ResNets. Next we compare networks within the same architectural family, first is a comparison between 2 TCNs with 4 and 8 Causal Blocks (12L and 24L) in Figure 6c. On the vertical axis are the layers for the larger TCN (24L) and the horizontal axis represents the smaller TCN(12L). We observe that the top half of TCN-24L shares alternating and intermittent similarity with some layers of TCN-12L, but as we go deeper in the network, this similarity fades away. This observation can also be reaffirmed for another set of TCNs in Figure 6g. Finally, we conduct a similar comparison between 2 ResNets with 6 and 10 Convolution Blocks, results of which are presented in Figure 6d. We observe the 2 ResNets learn very similar representations along their depths to a much higher degree when compared to two TCNs, like in Figure 6c. The only architectural difference between TCNs and ResNets is the use of dilated causal convolutions in TCNs. The presence of such convolutions ensures that each convolution block of a TCN parses its input as a unique causal sub-sequence, as opposed to time invariant convolutions in ResNets where each input and output of each block share a higher degree of temporal proximity with no regard to learning temporally plausible correlations.

3.5 Intra-Layerwise analysis of networks with epochs Next, Along the lines of SSC-CKA [25],

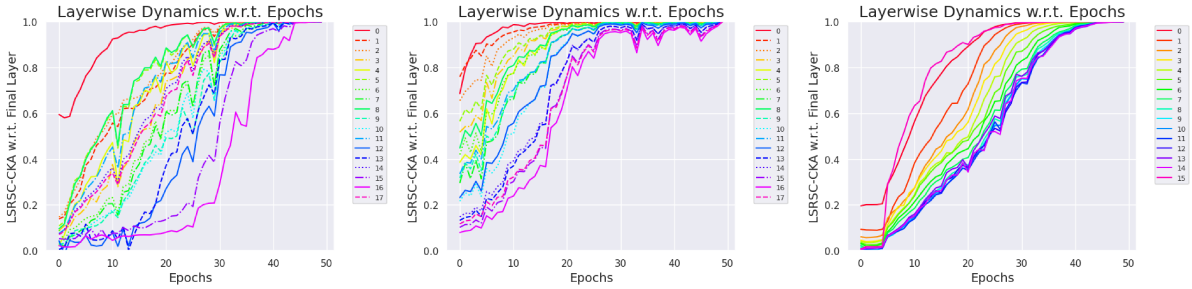


(a) TCN - 30 vs ResNet - 30. **(b)** TCN - 18 vs ResNet - 18. **(c)** TCN - 24 vs TCN - 12. **(d)** ResNet - 30 vs ResNet - 18.

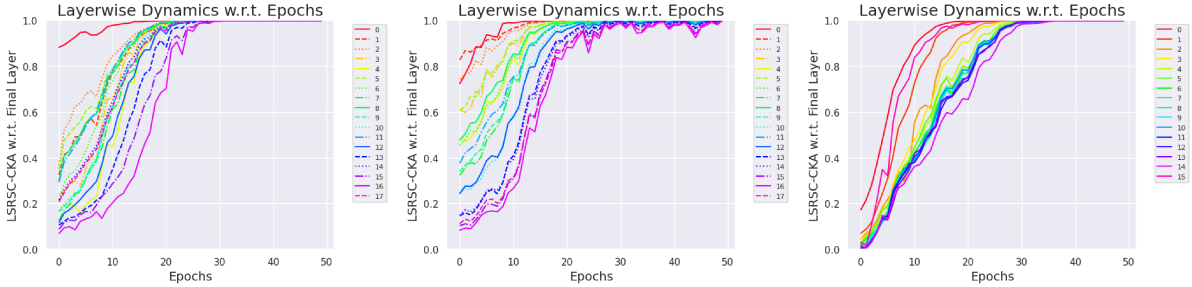


(e) TCN - 24 vs ResNet - 24. **(f)** TCN - 12 vs ResNet - 12. **(g)** TCN - 24 vs TCN - 18. **(h)** ResNet - 30 vs ResNet - 24.

Figure 6: Pairwise Network Analysis with LSRSC-CKA. Dataset - FordA. For each comparison plot the first network is represented by the vertical axis and the second network is represented on the horizontal axis. Networks within the same genre of operations display a higher similarity to each other than otherwise.



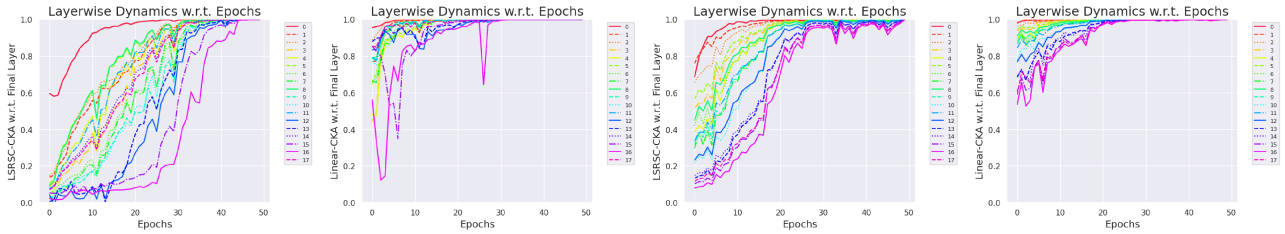
(a) LSRSC-CKA: TCN - 18 Layers. Network Accuracy - 95.60% **(b)** LSRSC-CKA: ResNet - 18 Layers. Network Accuracy - 93.78% **(c)** LSRSC-CKA: MLP - 15 Layers. Network Accuracy - 75.6%



(d) LSRSC-CKA: TCN - 18 Layers. Network Accuracy - 83.08% **(e)** LSRSC-CKA: ResNet - 18 Layers. Network Accuracy - 81.72% **(f)** LSRSC-CKA: MLP - 15 Layers. Network Accuracy - 63.82%

Figure 7: Intra-Layerwise LSRSC-CKA dynamics. Evaluation Dataset : Row 1 - FordA, Row 2 - FordB

PWCCA [18], we evaluate the behaviour epoch-wise of each layer with respect to its final epoch. The goal of this experiment to understand the training dynamics of all the layers of a network to get a better understanding of how the representations at each layer evolve as the training progresses and to observe any differences across different architectures. To conduct this analysis we compute the Affinity Coefficients based on algorithm 1 for every layer at all epochs. Then we compute the LSRSC-CKA similarity between a layer's Affinity Coefficients at every epoch with its Affinity Coefficients after the final epoch. Results for those experiments are



(a) LSRSC-CKA Layerwise convergence dynamics: TCN - 18 Layers. Network Accuracy - 95.60%

(b) Linear-CKA Layerwise convergence dynamics: TCN - 18 Layers. Network Accuracy - 95.60%

(c) LSRSC-CKA Layerwise convergence dynamics: ResNet - 18 Layers. Network Accuracy - 93.78%

(d) Linear-CKA Layerwise convergence dynamics: ResNet - 18 Layers. Network Accuracy - 93.78%

Figure 8: Intra-Layerwise dynamics. Evaluation Dataset : FordA. LSRSC-CKA offers better resolution between different layers of a network than Linear-CKA.

presented in Figure 7 for the FordA dataset. For each architecture we observe that shallower layers converge earlier than deeper layers, this phenomena was also observed in SSC-CKA[25], SVCCA[22], PWCCA[18]. However, we also notice that unlike ResNets in Figure 7b and MLPs in Figure 7c where this depth-wise convergence phenomena is self evident, Layers of TCNs do not strictly adhere to this phenomena. Instead in a TCN, Figure 7a, the layer convergence is fairly out of order, which indicates that earlier stages of a TCN continually evolve to learn newer representations at a much higher rate than ResNets and MLPs. As a final note we would also like to draw parallels between block structures and layer-wise convergence patterns, this is particular well demonstrated by looking at MLPs where deeper layers which constitute a block structure, shown in Figure 1c also are very similar in their convergence properties, as shown in Figure 7c.

We now use utilize this setup to show some interesting differences between LSRSC-CKA and Linear-CKA in terms of their ability to resolve between different layers of the network. In Figure 8a and Figure 8b we take a TCN trained on the FordA dataset and compare the LSRSC-CKA and the Linear-CKA[15, 20, 23, 21] respectively. We observe that while LSRSC-CKA and Linear-CKA both offer similar insights to a network's training dynamics. Linear-CKA is less adept at distinguishing between the dynamics of various layers. This is because Linear-CKA mostly influenced by the similarity between dominant principal components[7],[8] and LSRSC-CKA is approximately a scaled version of Linear-CKA with a more compressed singular value distribution [27]. This phenomena is further alluded to in Section 4 where we compare LSRSC-CKA with SSC-CKA[25] and Linear-CKA[15, 20, 23, 21].

4 Comparisons with Related works

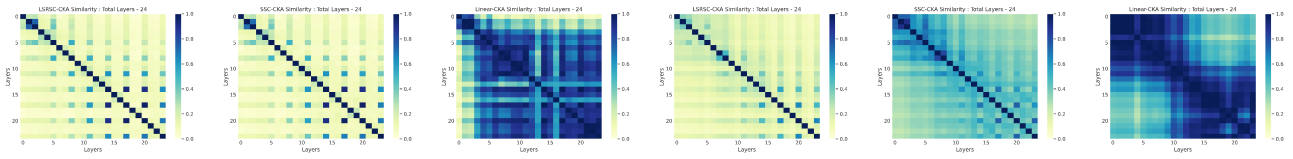
In this section we layout relevant works in the area of representation similarity based neural network analysis and compare our work with them. Priors works like Linear-CKA[15, 19, 24] and SSC-CKA[25] have been used to analyse the internal representational structure of various neural networks for the task of image classification. Our work inherits ideas from the literature of subspace clustering to improve upon Linear-CKA by incorporating higher order information than just pairwise dot products.Next, We compare LSRSC-CKA with Linear-CKA and SSC-CKA. To do so, We use a TCN with 8 Convolution Blocks and demonstrate the corresponding LSRSC-CKA, SSC-CKA and Linear-CKA on FordA and InsectWingbeatSound Dataset in Figure 9. Comparing LSRSC-CKA, Figure 9a and Figure 9d, with SSC-CKA, Figure 9b and Figure 9e, we observe that both approaches offer very similar layer-wise resolution and analytical insights, but since LSRSC-CKA can be solved analytically, it is orders of magnitude faster than SSC-CKA while being practically as useful as Linear-CKA as demonstrated in Table 1. Elaborating further,

Runtime of all analyzed methods in Seconds			
Method vs Dataset	LSRSC-CKA	SSC-CKA	Linear-CKA
FordA	22s	3782s	1s
InsectWingbeatSound	48s	288s	1.6s

Table 1: Runtime experiments for LSRSC-CKA, SSC-CKA and Linear-CKA. Architecture - Temporal Convolution Network. Dataset - FordA and InsectWingbeatSound

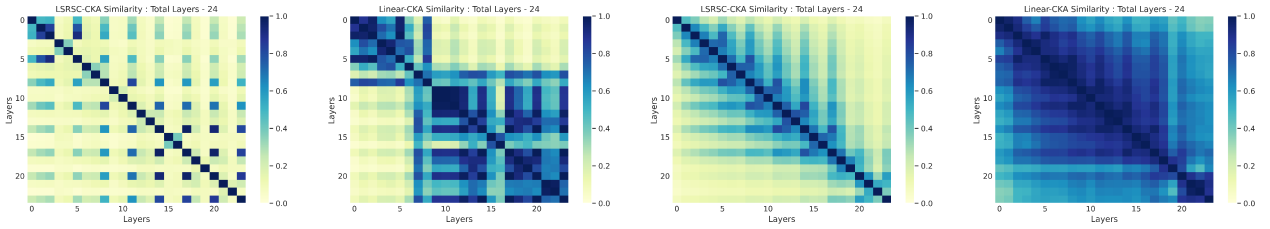
the reason for a lower order of magnitude difference between LSRSC-CKA and SSC-CKA on InsectWingbeatSound dataset is because SSC-CKA converges to a sub-optimal solution and stagnates with a relative error, Equation 2.3, higher than 0.2. Whereas the relative error in case of LSRSC-CKA was always below 0.01.

Next we compare LSRSC-CKA, Figure 9a and Fig-



(a) LSRSC-CKA **(b)** SSC-CKA **(c)** Linear-CKA **(d)** LSRSC-CKA **(e)** SSC-CKA **(f)** Linear-CKA

Figure 9: Comparison between LSRSC-CKA, SSC-CKA and Linear-CKA. Network - TCN : 24 Layers. Figure 9a-Figure 9c contain the results for FordA and Figure 9d-Figure 9f for IWB



(a) LSRSC-CKA of TCN 24 Layer **(b)** Linear-CKA of TCN 24 Layer **(c)** LSRSC-CKA of ResNet 24 Layer **(d)** Linear-CKA of ResNet 24 Layer

Figure 10: LSRSC-CKA vs Linear-CKA Resolution Comparison. Networks - TCN 24 Layer and ResNet 24 Layer. Dataset - FordB. We Also observe that corresponding Linear-CKA analysis offers a lower resolution of layer wise similarity.

ure 9d, with Linear-CKA, Figure 9c and Figure 9f. A notable phenomena that we observe when comparing the two is the lower resolution of Linear-CKA in discerning layers within a block-diagonal structure. While both methods capture the presence of block structures, the resolution at which they do so are very different.

Continuing the analysis on FordB dataset, we compare LSRSC-CKA with Linear-CKA in terms of their ability to resolve between different layers of the network, the results of which are presented in Figure 10. When comparing LSRSC-CKA, Figure 10a and Figure 10c, with Linear-CKA, Figure 10b and Figure 10d, we again observe that while both algorithms capture the macro block-diagonal structures, Linear-CKA has a poorer resolution in terms of distinguishing different layers within from each other. This is true both for Temporal Convolution Networks and ResNets, as shown in Figure 10. This lack of resolution by Linear-CKA is detrimental to its application in sparse data domains, as it tends to converge to a macro block-diagonal associated with generally over-parameterized networks even when there might be unique micro-structures along the block diagonal that maybe architecture specific, as captured by LSRSC-CKA and SSC-CKA.

5 Conclusions

With works like [28] having laid the ground for the application of deep learning methods for time series analysis. This has opened the door for creating interesting network architectures like TCN[3] tailored for problems in time series mining. Such a development poses an interesting question - Do networks like TCNs work the same way as ResNets? In thus study we answer this question by relying on subspace clustering and representation similarity techniques. We utilize many datasets from UCR Time Series Archive[6] and show that Causal Convolutions helps TCNs learn newer representations with the addition of layers. The effect of these leads to TCNs not prominently exhibiting the block-structure[15, 19, 25] in its layer-wise similarity plots. We also investigate the behaviour of internal layers of each of these networks by comparing them through different stages of training. We observe that for Acausal Networks, shallower layers of the network convergence much faster to their final state than the deeper ones, this behaviour has also been observed in Deep Networks for Image Classification [25, 22, 18]. However Temporal Convolution Networks don't strictly exhibit such a depth dependent convergence phenomena. Instead even some shallower layers keep learning newer representations with epochs. Our study is among the first empirical works to analyze neural networks in the domain of time series classification and offers a convenient tool to better understand the effects of different Inductive Priors on Networks in time series data.

6 Acknowledgements

This work was done during the first author's internship at VISA Research and UCR co-authors were also partly supported by the National Science Foundation CREST Center for Multidisciplinary Research Excellence in Cyber-Physical Infrastructure Systems (MECIS) grant no. 2112650. Any opinions, findings, and conclusions or recom-

mendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] Anthony Bagnall, Aaron Bostrom, James Large, and Jason Lines. The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version, 2016.

[2] Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with cote: The collective of transformation-based ensembles.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[4] Mustafa Gokce Baydogan, George C. Runger, and Eugene Tuv. A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:2796–2802, 2013.

[5] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *CoRR*, abs/1203.0550, 2012.

[6] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive, 2018.

[7] MohammadReza Davari, Stefan Horoi, Amine Natik, Guillaume Lajoie, Guy Wolf, and Eugene Belilovsky. Reliability of cka as a similarity measure in deep learning, 2022.

[8] Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding representation similarity with statistical testing, 2021.

[9] Philippe Esling and Carlos Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1), dec 2012.

[10] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *CoRR*, abs/1809.04356, 2018.

[11] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Algorithmic Learning Theory*, pages 63–77, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[12] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[14] Eamonn Keogh. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases*, VLDB '02, page 406–417. VLDB Endowment, 2002.

[15] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited, 2019.

[16] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and knowledge discovery*, 15(2):107–144, 2007.

[17] Can-Yi Lu, Hai Min, Zhong-Qiu Zhao, Lin Zhu, De-Shuang Huang, and Shuicheng Yan. Robust and efficient subspace segmentation via least squares regression, 2014.

[18] Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation, 2018.

[19] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *CoRR*, abs/2010.15327, 2020.

[20] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth, 2020.

[21] Thao Nguyen, Maithra Raghu, and Simon Kornblith. On the origins of the block structure phenomenon in neural network representations, 2022.

[22] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017.

[23] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks?, 2021.

[24] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *CoRR*, abs/2108.08810, 2021.

[25] Uday Singh Saini, Pravallika Devineni, and Evangelos E. Papalexakis. Subspace clustering based analysis of neural networks. *CoRR*, abs/2107.01296, 2021.

[26] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.*, 29(6):1505–1530, nov 2015.

[27] Rene Vidal. Attention: Self-expression is all you need, 2022.

[28] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR*, abs/1611.06455, 2016.