Supplementary Materials

Conic Activation Functions

A Hyperbolic Geometry

Definition A.1 (Minkowski)**.** *A point (called an event) x is defined in the C-dimensional Euclidean space (called space-time). A scalar product on $\mathbb{R}^C$ is defined as*

$$\langle x, y\rangle_M = x_1 y_1 - x_2 y_2 - \ldots - x_C y_C \tag{8}$$

The hyperbolic geometry can be understood by the fact that along a rotation in the space, the quantity $x_1^2 - x_2^2 - \ldots x_C^2$ is unchanged. This scalar product induces a norm $|x|_M = \sqrt{\langle x, x\rangle_M}$, and the Lorentz cone is defined as $V = \{x \in \mathbb{R}^C : |x|_M \geq 0, x_1 \geq 0\}$. It is usually called a *light cone* since if we regard $x_1 := t$ as the time axis where the constant c is the speed of light and t is the time of the event x, then the cone is characterized by $\sqrt{|x_\perp|} = ct$, and c is the tangent value of the opening angle of the cone, and we set $c = 1$ without loss of generality. More precisely it is a future light cone since $t \geq 0$, and the past light cone associates to the case when $t \leq 0$. CoLU sets the past light cone with $c = 0$. The *plane of simultaneity* (under the rest frame of reference) is defined as $H(x) = \pi_1^{-1}(x_1 e_1) = \{y \in \mathbb{R}^C : y_1 = x_1\}$. In Figure 1a, CoLU is intuitively understood as the closest point to the input within the light cone and the plane of simultaneity. The meaning of the weight w after the activation function is visualized in Figure 1b, where the previous space-time is tilted by a linear transform (called Lorentz transform). In the grouped CoLU case, gluing the axes together is motivated by equalizing the time axes of each light cone (associated with an observer).

B Relation with Noether's Theorem

In this section, we associate the CoLU equivariance with the conserved quantity in the fiber space indexed by the spatial domain and show that CoLU and self-attention have the same type of symmetry.

Definition B.1 (Lagrangian)**.** *A Lagrangian functional is defined as an integral $\mathcal{L} : TM \longrightarrow \mathbb{R}$ such that*

$$\mathcal{L}(x, \dot{x}, \mathrm{L}) = \int_0^L \mathrm{L}(x, \dot{x}, \ell)\, \mathrm{d}\ell \tag{9}$$

Theorem B.2 (Noether)**.** *Suppose $\forall s \in \mathbb{R}$ the Lagrangian $\mathcal{L}(x, \dot{x}, L)$ is invariant over a transformation h^s parameterized by s, then the following quantity is constant over time.*

$$I = \frac{\mathrm{dL}}{\mathrm{d}\dot{x}} \frac{\mathrm{d}h^s}{\mathrm{d}s} \tag{10}$$

Corollary B.3 (Translation Momentum)**.** *Assume $\omega \in \Omega = [-1, 1]^2$, $e_1 = (1, 0)$ is a unit vector, and $\mathrm{L}(\omega, \dot{\omega}, t) = \dot{\omega}^2/2$. If $h^s(\omega) = \omega + s e_1$ then $I = \dot{\omega}_1$ is conserved.*

The convolution function commutes with h^s and associates with the translation momentum on Ω.

Corollary B.4 (Angular Momentum)**.** *Assume $x \in \mathbb{R}^\Sigma$ with $\Sigma = \{1, 2, 3\}$,$|\Sigma| = C = 3$, $e_2, e_3 \in \mathbb{R}^C$ are unit vectors of starting and ending directions of a rotation R around e_1. If $h^s : x(\sigma) \in \mathbb{R}^\Sigma \mapsto R^{2s/\pi} x(\sigma)$, then $I = \dot{x} \times e_1$.*

Proposition B.5 (Attention Invariance)**.** *The self-attention function commutes with h^s, so the Lagrangian of attention dynamics admits the orthogonal group. Therefore the attention dynamics in Equation (39) conserves the angular momentum for rotations in $\mathbb{R}^C$.*

Proposition B.6 (Conic-Activation Invariance)**.** *For the same reason as above, if the activation function is conic, The ResNet dynamics in Equation (40) conserves the angular momentum for rotations around the cone axis.*

C Relation with Linear Mode Connectivity

The equivariance of activation functions is linked to the linear mode connectivity phenomenon: two neural networks trained with different initializations and (usually) on the same dataset can be aligned to be very close to each other (Izmailov et al., 2018; Singh & Jaggi, 2020; Entezari et al., 2022; Ainsworth et al., 2023). This phenomenon implies that neural network optimization is approximately convex modulo a group. The group characterizes the permutation symmetry of component-wise activation functions, and the proposed conic activation functions generalize the type of symmetry. This aligned representation phenomenon across different models at a larger scale is discussed in (Huh et al., 2024). Note that there are other types of mode connectivity (Garipov et al., 2018), which does not leverage permutation symmetry and requires more complicated paths such as piece-wise linear or Bézier spline, and we do not discuss here.

Given the loss function $L(\theta)$ on two sets of model parameters θ_0, θ_1, the closeness of the two models is measured by the loss barrier. There are different definitions of loss barriers, and we define it as

$$\sup_{s\in[0,1]} B_{\theta_0,\theta_1}(s) = L((1-s)\theta_0 + s\theta_1)/((1-s)L(\theta_0) + sL(\theta_1))) - 1 \tag{11}$$

The loss barrier signifies the relative loss increase of the linearly interpolated weights. With one model θ_0 fixed, an alignment on the other one θ_1 refers to finding the optimal permutation on each layer by matching either intermediate states or weights (Jordan et al., 2023; Ainsworth et al., 2023). The proposed activation function generalizes permutations to orthogonal matrices (where permutations are special cases). The orthogonal symmetry is continuous, meaning that there are infinitely many ways of alignment. This results in a loss landscape with infinite local minima forming connected components. The alignment matrices are associated with different manifold constraints.

D More Experiments

D.1 Toy VAE

Experimental Settings The VAE's encoder and generator's parameters are $\theta_{\mathrm{E}} = (w_{\mathrm{E}}, w'_{\mathrm{E}})$ and $\theta_{\mathrm{G}} = (w_{\mathrm{G}}, w'_{\mathrm{G}})$. The inputs, latents and outputs are $x, z, \widehat{x}$, where $z = w_{\mathrm{E}}\lambda(w'_{\mathrm{E}}x)$ and $\widehat{x} = w_{\mathrm{G}}\lambda(w'_{\mathrm{G}}z)$. The dimension of input and output is $28 \times 28 = 784$ and the dimension of the hidden state z is fixed to $d = 20$. The loss function is defined as

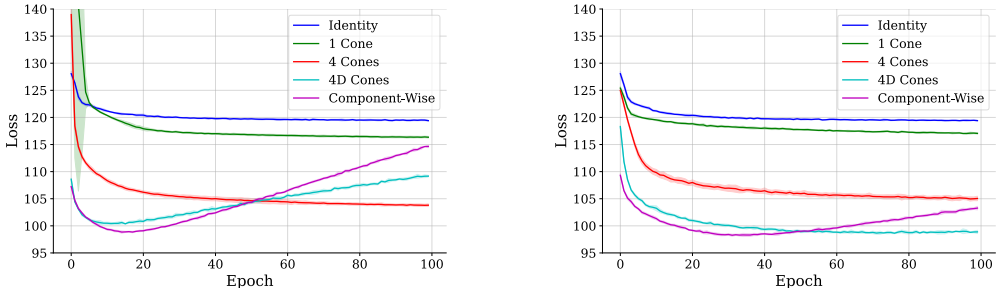
$$L(\theta) = H(x, \widehat{x}) + \alpha \mathcal{D}_{\mathrm{KL}}(p_z|p_0) \tag{12}$$

where $H(x, \widehat{x}) = -\sum_n x_n(\log(\widehat{x_n}) + (1 - x_n)\log(1 - \widehat{x_n}))$ is the binary cross-entropy, and $\mathcal{D}_{\mathrm{KL}}(p_z|p_0)$ is the Kullback-Leibler Divergence from a standard Gaussian distribution $p_0 \sim \mathcal{N}(0, 1)$ to the latent distribution $p_z \sim \mathcal{N}(\mu_z, \sigma_z)$

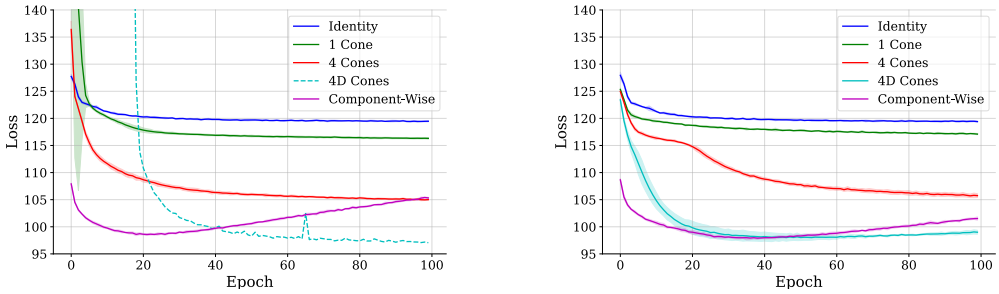
$$\mathcal{D}_{\mathrm{KL}}(p_z|p_0) = -\int_x p_0(x)\log(p_z(x)/p_0(x))\,\mathrm{d}x = \frac{1}{2}\sum_{j=1}^{d}\left(1 + \log({\sigma_z}_j^2) - {\mu_z}_j^2 - {\sigma_z}_j^2\right) \tag{13}$$

The last equality is obtained by setting $\mu_z = (\sum_{n=1}^{N} z_n)/N$ and $\sigma_z = ((\sum_{n=1}^{N}(z_n - \mu_z)^2)/(N - 1))^{1/2}$ with sample size N. The hyperparameter α is set to 1 so that the impact of the KL term is relatively small, given that the magnitude of the cross-entropy term is around 100 times larger.

More Results Figure 8 visualizes the test loss curves when the granularity of grouping varies. In summary, as the cone dimension S reduces, the performance of grouped conic activation functions improves until it outperforms component-wise activation functions. In the figures, only one case for high and low dimensional cones is shown for clarity.The cone dimension is among $S \in \{\infty, 2400, 600, 4, 2\}$, or equivalently, the number of groups is among $G \in \{0, 1, 4, 800, 2400\}$. In the shared-axis case, the network width is fixed to $C = 2401$ so that the number of parameters is the same. In the no-sharing case, the network width is $C = 2400 + G$. Specifically, $G = 0$ reduces to the identity function, while $S = 2$ (or $G = 2400$) is specified as the component-wise activation of ReLU for hard projection and SiLU for soft projection since there is no orthogonality. The dashed line in the hard-projected shared-axis case means that the training is unstable over different random seeds: about 80% of the initializations do not converge, so only one converged training instance is



(a) Hard (b) Soft



(c) Hard Shared (d) Soft Shared

Figure 8: Test loss curves of a VAE with two-layer encoder and decoder with standard deviation regions. The left and right figures correspond to hard and soft projections, and the top and bottom correspond to hard projection and soft projection.

visualized. Sharing the axes increases the performance of the activation function on toy examples when the projection is soft and the cones are low dimensional (S being small). In practice, this combination is the most meaningful one since it has the best performance and saves the most number of parameters. Intuitively, soft projection effectively stabilizes the training of CoLU models, which is the most obvious in the early training stage of the highest-dimensional conic functions (the single cone case). Especially, it makes the VAE with shared-axis activation functions easier to train.

D.2 Toy MLP

Experimental Settings The model is parameterized by $\theta = (w, w')$ and defined as $x \in \mathbb{R}^{28\times 28} \mapsto \widehat{y} = \mathrm{softmax}(w\lambda(w'x)) \in \Delta^9$, which is a two-layer MLP whose output is mapped to the probability simplex by a softmax function. The MNIST dataset is denoted as a collection of data pairs (x, y), where x is flattened as vectors and y is a unit vector among 10 classes. The network width is fixed to $C = 512$. The loss function is the cross entropy of the predicted probability relative to the label

$$H(\widehat{y}, y) = \sum_i y_i \log \widehat{y}_i \tag{14}$$

D.3 Diffusion Models

Training Experimental Settings The UNet structure follows the Stable Diffusion model (LDM) (Rombach et al., 2022) without the VAE part. The network block widths are set to $(128, 256, 256, 256)$ and the numbers of ResNet blocks are set to 1 for CIFAR10 (2 for Flowers). For unconditional generation, the cross-attention function is replaced with the self-attention function. All runs last 100K steps and use the Adam optimizer with a batch size of 128 for CIFAR10 (16 for Flowers), a learning rate of 10^{-4}, and a weight decay coefficient of 10^{-2}. Figure 9 shows comparisons on the Flowers dataset.

(a) ReLU (b) CoLU

Figure 9: Samples of diffusion models trained on the Flower Dataset.

Figure 10: More CoLU text-to-image samples.

Fine-Tuning Experimental Settings The pretrained model has 835 million parameters and is trained on the LAION dataset. The architecture is identical to the Stable Diffusion model with block width $(320, 640, 1280, 1280)$. The training details are the same as above. The pre-trained SiLU model and the text-to-image Pokémon dataset are from the diffusers library (von Platen et al.). Figure 11 visualizes the comparisons between a fine-tuned SiLU model and a fine-tuned soft CoLU

model with the same text prompt and initial noise in the diffusion model. Figure 10 shows more samples of the fine-tuned model with text prompts generated by a large language model.

Figure 11: LDM samples of a fine-tuned Soft CoLU model and a fine-tuned SiLU model.

D.4 MLP in GPT2

Experimental Settings The Transformer follows Vaswani et al. (2017) with a block size of 64, an embedding size of 256, a number of heads 8, head size 32 and number of layers 6 (for small setting). For large setting, we set a block size of 64, an embedding size of 768, a number of heads 12, a head size of 64, and a number of layers 12. Each run lasts 20K steps and uses the Adam optimizer with a batch size of 512, a learning rate of 10^{-4}, and a weight decay coefficient of 10^{-2}.

D.5 Linear Mode Connectivity

The latent state's permutation symmetry is studied qualitatively on diffusion UNets and quantitatively on toy models.

Convolution Filter Symmetry We train individual diffusion UNets on the CIFAR10 dataset with different random seeds and qualitatively show that the palette filters (the last convolution layer of the generative model) in a ReLU-model can be permuted to match each other, whereas a CoLU-model cannot, showing that the symmetric pattern is essentially different from permutation. The diffusion model implementation is based on (Salimans & Ho, 2022) and we only change the activation function to be conic with $G = 32$ without axis sharing. We take a global batch size of 128 and a learning rate of 10^{-4}. After around 5K steps the generated images are perceptually visible. Figure 15 visualizes the last convolution layer w (which we call a palette) of dimension $256 \times 3 \times 3 \times 3$ in SiLU model and soft CoLU model, each with two different initializations. The colors are linearly scaled for better visualization. The left two sets of filters can be permuted to match each other, whereas the right two sets cannot since they are orthogonal symmetric except for the axes. We observe that the

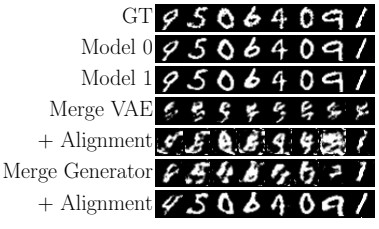


Figure 12: Random samples of ground truth, VAE outputs, merged VAE, and merged generator with fixed latents.

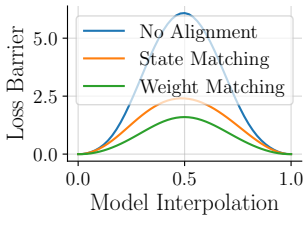


Figure 13: Loss barriers between the aligned models by state or weight matching.

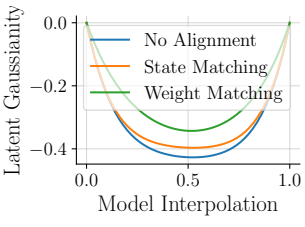
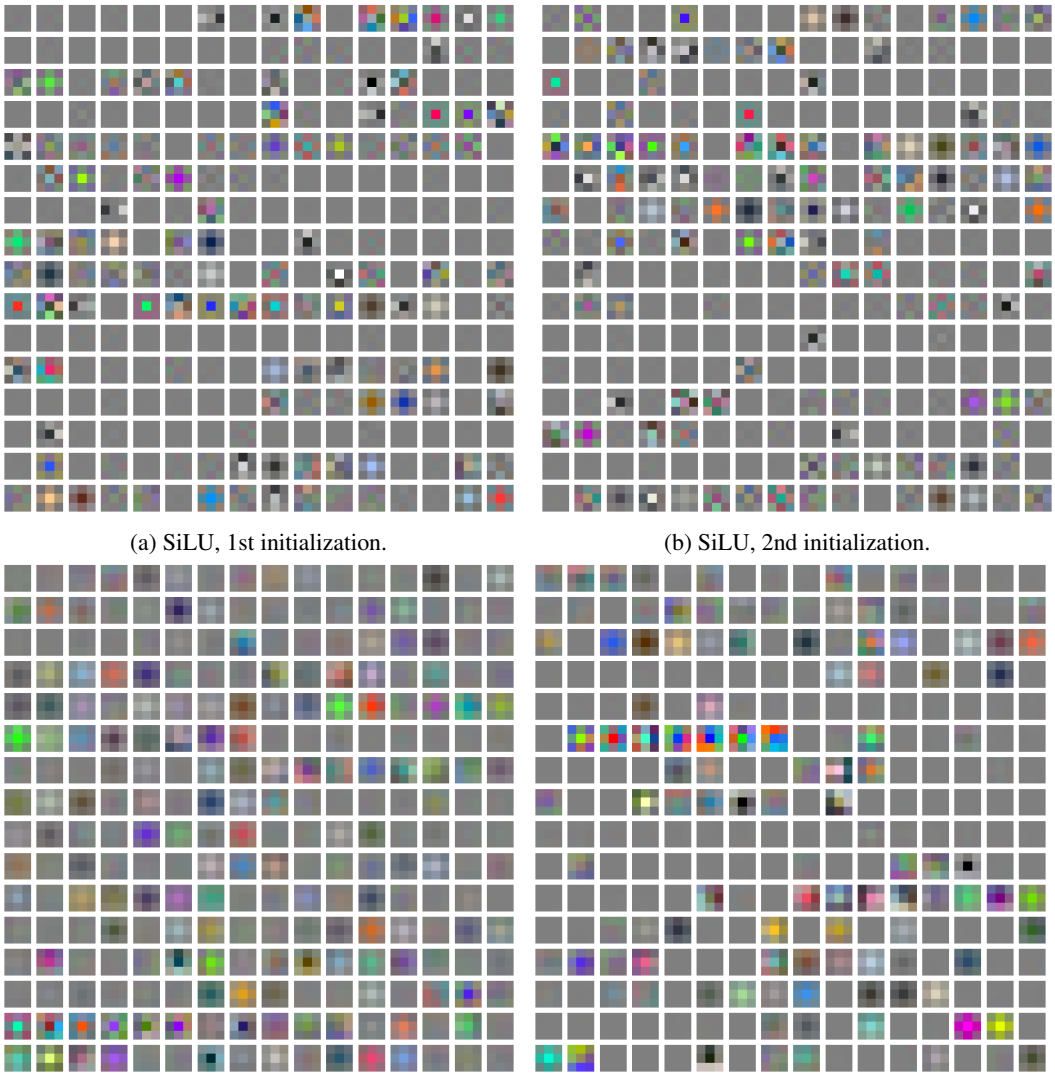


Figure 14: KL Divergence between $\mathcal{N}(\mu_z, \sigma_z)$ and $\mathcal{N}(0, 1)$ on the interpolation paths.



(a) SiLU, 1st initialization. (b) SiLU, 2nd initialization.

(c) soft CoLU, 1st initialization. (d) soft CoLU, 2nd initialization.

Figure 15: Palettes of diffusion models with SiLU and soft CoLU. The first row can be permuted to match each other whereas the second row cannot.

last layer has more visually plausible patterns than the first layer in the denoising UNet, different from most works in the literature do for recognition models.

Generative Model Alignment We show linear mode connectivity results for the same toy model in Section 5.2, and we find out that linear mode connectivity also holds in generative models, which is rarely discussed in the literature.

Weight matching and state matching algorithms in Appendix F are applied to align the VAE model, and the results are shown in Figure 12. They have different advantages: weight matching produces a flatter barrier in our toy experiment, and state matching requires no data as the model input. Their convergence is analyzed in (Ainsworth et al., 2023; Jordan et al., 2023). The difference in the conic case is that the symmetry group is relaxed, so the Stiefel manifold optimization problem replaces the sum of bilinear assignment problem (SOBLAP). Figure 13 and 14 visualize the loss barrier and the KL Divergence barrier.

E Proofs

Proof of Proposition 4.2. If $|x_\perp| \neq 0$, Equation (6) holds component-wise, and the set $\{x \in \mathbb{R}^C : |x_\perp| = 0\}$ is negligible. □

Proof of Lemma 4.5. We assume $P \in \mathrm{GA}(C)$ To prove $\mathcal{G}_\lambda \subset \mathcal{G}_{\mathcal{I}_\lambda}$, it suffices to show $\forall P \in \mathcal{G}_\lambda, \mathcal{I}_\lambda = PP^{-1}\mathcal{I}_\lambda = P\mathcal{I}_\lambda \subset I_\lambda$. The last inclusion comes from $\forall P \in \mathcal{G}$, there holds $\forall x \in \mathcal{I}_\lambda, \lambda(Px) = P\lambda(x) = Px$, so $Px \in \mathcal{I}_\lambda$. The first equality is from $P \in \mathcal{G}_\lambda$ and the second one is from $x \in \mathcal{I}_\lambda$. Conversely, to prove $\mathcal{G}_{\mathcal{I}_\lambda} \subset \mathcal{G}_\lambda$, we need to strengthen the condition on λ to $\exists A$ a convex set such that $\forall x, \lambda(x) = \mathcal{P}_A(x)$. $\forall z \in \mathcal{I}_\lambda, \langle z - P\lambda(x), Px - P\lambda(x)\rangle \geq 0$, so $\lambda(Px) = P\lambda x$ □

Proof of Proposition 4.8. (1) is proven by taking x and P such that

$$w'x = (1, 0, 0, \ldots, 0), \quad P[1, 2; 1, 2] = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

(2) is proven by taking

$$w^\dagger = \begin{bmatrix} 0 & \mathbf{I}_2 \\ \mathbf{I}_{C-2} & 0 \end{bmatrix}, \quad w'^\dagger = \begin{bmatrix} 0 & \mathbf{I}_{C-2} \\ \mathbf{I}_2 & 0 \end{bmatrix}$$

□

Proof of Remark 4.9. It suffices to take η large enough so that $D \in \mathrm{Diag}(C)$ is determined by $\mathrm{argmin}_{PD \in \mathcal{G}_\lambda} \|PD\theta\|$, since $P \in \mathrm{Perm}(C)$ does not change $\|P\theta\|$. □

F Algorithms

Algorithm 1 and 2 from Jordan et al. (2023); Ainsworth et al. (2023) are applied to achieve linear mode connectivity of the toy VAE model.

Algorithm 1 Weight Matching

Require: θ_0, θ_1 ▷ Pre-Trained Weights from different random initializations
Require: $x \in X$ ▷ Intermediate states ordered by forward pass
Require: $\theta_{\text{prev}}(x), \theta_{\text{next}}(x)$ ▷ Linear weights prior to and after the state
Require: $\mathcal{G}$ ▷ Symmetry group of the activation function
Ensure: $P = \{P_x : x \in X\}$ ▷ Optimal alignment
Initialize $P_x = \mathbf{I}_{\dim(x)}$ ▷ Identity matrices with the same dimension of x
repeat
  for x in $\mathrm{RandPerm}(X)$ **do** ▷ Shuffle the order of the states
    $L(P) = 0$
    for w' in $\theta_{\text{prev}}(x)$ **do**
      for w in $\theta_{\text{next}}(x)$ **do**
        $L(P) \leftarrow L(P) + \mathrm{tr}({w'_0}^\top P w'_1)/|\theta_{\text{prev}}(x)| + \mathrm{tr}(w_0 P^\top w_1^\top)/|\theta_{\text{next}}(x)|$
      end for
    end for
    Solve $P_x \leftarrow \mathrm{argmin}_{P \in \mathcal{G}} L(P)$
  end for
until P Converges

G Unification of Neural Networks

This section aims to establish a bottom-up framework from first principles to infer the form of neural network architectures, including the proposed activation function. For simplicity, we assume that each state is defined in a vector space with a fixed dimension $M = \mathbb{R}^C$. We separate the construction into several parts, including a general Neural Network, a Residual Network, a Convolutional Network, and an Attention Network.

Algorithm 2 State Matching

Require: $\theta_0, \theta_1, \theta_{\text{prev}}(x), \theta_{\text{next}}(x), \mathcal{G}$ ▷ Same as above
Require: $x(0)$ ▷ Data as model input
Require: $x \in X(\theta, x(0))$ ▷ Following the order of the forward pass
Ensure: $P = \{P_x : x \in X(\theta, x(0))\}$ ▷ Optimal alignment

```
Initialize P_x = I_{dim(x)}
for (x_0, x_1) in (X(θ_0, x(0)), X(θ_1, x(0))) do
    Solve P_ℓ ← argmin_{P∈G} L(P) = x_0^⊤ P x_1
    for w' in θ_prev(x) do
        w_1 ← P_ℓ w_1
    end for
    for w in θ_next(x) do
        w_1 ← P_ℓ w_1
    end for
end for
```

Proposition G.1 (Derivation of a Neural Network)**.** *The assumptions on the left of the following equations characterize the neural network in Equation (20).*

$$x(1) = \Lambda(x(0)) \tag{15}$$

$$\xrightarrow{\textit{Process Decomposition}} x(L) = \Lambda_L \Lambda_{L-1} \dots \Lambda_1(x(0)) \tag{16}$$

$$\xrightarrow{\textit{Linear Kernel Space}} x(L) = w(L)\Lambda_L(w'(L) \dots w(1)\Lambda_1(w'(1)(x(0)) \dots)) \tag{17}$$

$$\xrightarrow{\textit{Time Homogeneity}} x(L) = w(L)\Lambda(w'(L) \dots w(1)\Lambda(w'(1)(x(0)) \dots)) \tag{18}$$

$$\xrightarrow{\textbf{\textit{Component-Wise}}} x(L) = w(L)\lambda(w'(L) \dots w(1)\lambda(w'(1)(x(0)) \dots)) \tag{19}$$

$$\xleftrightarrow{\textit{Iterative Form}} x(\ell) = w(\ell)\lambda(w'(\ell)x(\ell-1)), \ell = 1, 2, \dots L \tag{20}$$

In the derivation above, equation (15) denotes an arbitrary function Λ with input $x(0)$ and output $x(1)$. Equation (16) holds by assuming the function decomposes into several ones, resulting in a process or a sequence of states $x(0), x(1), \dots, x(L) \in M$, where the terminal states $x(0)$ and $x(L)$ are the input and output. Equation (17) follows from assuming the sequence of functions to perform in a linear kernel space. Suppose the linear kernel function at layer ℓ parameterized by w'

$$\begin{aligned} \Phi : M &\longrightarrow M_\lambda \\ x &\longmapsto w'(\ell)x \end{aligned}$$

on the kernel space, there is a function $\Lambda : M_\lambda \to W_\lambda$ where W_λ is the range of the activation function. Then the inverse kernel function is parameterized by w

$$\begin{aligned} \widehat{\Phi} : W_\lambda &\longrightarrow M \\ x &\longmapsto w(\ell)x \end{aligned}$$

Again we assume for simplicity that the dimensionality of each kernel space is fixed: $M_\lambda = M = \mathbb{R}^C$. Equation (17) is obtained by replacing $x \in M$ with $x' \in M_\lambda$ in equation (16) and plugging in the change of variable $x' = wxw'$. Equation (18) is obtained by assuming *time homogeneity modulo a linear group* of the nonlinear functions: the function Λ is on the lifted space M_λ in Equation (17) instead of M, where the lifting is determined by assuming that there exist proper w, w' in each space such that the functions are uniform over time, meaning $\Lambda_1 = \Lambda_2 = \dots = \Lambda_L = \Lambda$. Equation (19) assumes that there exists a function $\lambda : \mathbb{R} \to \mathbb{R}$ so that the nonlinear function is represented as $\Lambda(x_1 e_1 + \dots + x_n e_n) = \lambda(x_1)e_1 + \dots + \lambda(x_n)e_n$. In this paper, we replace this assumption with orthogonal symmetry instead. Note that the component-wise $\lambda : M_\lambda \to M_\lambda$ is equivariant under any permutation P. Equation (20) rewrites the process into steps between adjacent states.

Proposition G.2 (Derivation of a Residual Network)**.** *Adding more assumptions, we continue to derive the form of a Residual Network in Equation (27).*

$$\overset{\textit{Linear Splitting}}{\Longleftrightarrow} \quad x(\ell) = \lambda(w'(\ell)x(\ell-1)) + (w(\ell)-1)\lambda(w'(\ell)x(\ell-1)) \tag{21}$$

$$\overset{\textit{Re-Parameterization}}{\Longrightarrow} \quad x(\ell) = \lambda(w'(\ell)x(\ell-1)) + w(\ell)\lambda(w'(\ell)x(\ell-1)) \tag{22}$$

$$\overset{\textit{Linear Branching}}{\Longrightarrow} \quad x(\ell) = \lambda(w''(\ell)x(\ell-1)) + w(\ell)\lambda(w'(\ell)x(\ell-1)) \tag{23}$$

$$\overset{\textit{Nonlinear Branching}}{\Longrightarrow} \quad x(\ell) = \lambda'(w''(\ell)x(\ell-1)) + w(\ell)\lambda(w'(\ell)x(\ell-1)) \tag{24}$$

$$\overset{w''=1}{\Longrightarrow} \quad x(\ell) = \lambda'(x(\ell-1)) + w(\ell)\lambda(w'(\ell)x(\ell-1)) \tag{25}$$

$$\overset{w'=1}{\Longrightarrow} \quad x(\ell) = \lambda'(x(\ell-1)) + w(\ell)\lambda(x(\ell-1)) \tag{26}$$

$$\overset{\textit{Residualization}}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + w(\ell)\lambda(x(\ell-1)) \tag{27}$$

In the derivation above, Equation (21) splits the inverse kernel function's weight w into the identity (zeroth-order) part and the first-order part $w-1$. Equation (22) re-parameterize the weights by denoting $1-w$ as w without loss of generality. Equation (23) modifies the assumption in Equation (17) so that two copies of kernel functions are parameterized by w'', w', and the inverse kernel function remains the same. Equation (24) modifies the assumption in equation (18) different functions λ', λ applies on each one. Equation (25) assumes that the first kernel function w'' is identity. Equation (26) further assumes w' is identity to simplify equations in the sequel. Equation (27) assumes that the function associating to the zeroth-order kernel space is identity.

Proposition G.3 (Derivation of a Convolutional Network)**.** *Given a basic neural network, the form of a convolutional neural network in Equation (33) is determined by the following additional assumptions on the left.*

$$\overset{\textit{Space Indexation}}{\Longleftrightarrow} \quad x(\ell) = x(\ell-1) + w(\ell,\omega,\omega',\sigma,\sigma')\lambda(x(\ell-1,\omega',\sigma')) \tag{28}$$

$$\overset{\textit{Summation Form}}{\Longleftrightarrow} \quad x(\ell) = x(\ell-1) + \sum_{\omega'\in\Omega} w(\ell,\omega,\omega',\sigma,\sigma')\lambda(x(\ell-1,\omega',\sigma')) \tag{29}$$

$$\overset{\textit{Equivariance}}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + \sum_{\omega'\in\mathbb{Z}^2} w(\ell,\omega'-\omega,\sigma,\sigma')\lambda(x(\ell-1,\omega',\sigma')) \tag{30}$$

$$\overset{\textit{Change of Variable}}{\Longleftrightarrow} \quad x(\ell) = x(\ell-1) + \sum_{\omega'\in\mathbb{Z}^2} w(\ell,\omega',\sigma,\sigma')\lambda(x(\ell-1,\omega'+\omega,\sigma')) \tag{31}$$

$$\overset{3\times 3\ \textit{Window}}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + \sum_{\omega\in\{-1,0,1\}^2} w(\ell,\omega+\omega',\sigma,\sigma')\lambda(x(\ell-1,\omega',\sigma')) \tag{32}$$

$$\overset{\textit{Convolution Notation}}{\Longleftrightarrow} \quad x(\ell) = x(\ell-1) + w(\ell)\star\lambda(x(\ell-1)) \tag{33}$$

In the derivation above, Equation (28) stacks the states of dimension $n = CHW$ into a tensor whose space dimensions is indexed by $\omega \in \Omega = [H]\times[W] \subset \mathbb{Z}^2$ (with the bracket notation $[n] = \{1, 2, \ldots, n\}$) and the channel dimension indexed by $\sigma \in [C]$. Equation (29) write the matrix-vector product in the form of a summation. In Equation (30) we imposes core assumption of the Convolutional Neural Network, namely the spatial translation equivariance, so that $(wx)(\omega-\omega'') = (wx(\omega-\omega'')), \forall\omega''$. This results in $w(\omega,\omega'+\omega'') = w(\omega-\omega'',\omega'), \forall\omega''$, so $w(\omega,\omega')$ must take the form of $w(\pm\omega\mp\omega')$, and we set $w(\omega'-\omega)$ without loss of generality. Equation (31) is a change of variable, replacing $\omega'-\omega$ with ω'. Equation (32) imposes the condition that the spatial dependency on Ω is within a 3×3 neighbourhood. Note that the family of neighbourhoods defines the *Topology* of the space Ω. Finally, Equation (33) denotes the linear function with the $\star$ notation.

Proposition G.4 (Derivation of an Attention Network)**.** *The construction of the cross-attention function is proceeded by imposing further assumptions.*

$$\overset{1\times 1 \text{ Window}}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + \lambda(x(\ell-1,\omega,\sigma'))w(\ell,\sigma',\sigma) \tag{34}$$

$$\overset{\text{Condition } k^Tk}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + \lambda(x(\ell-1,\omega,\sigma'))k(\ell,\sigma'',\sigma')^Tk(\ell,\sigma'',\sigma')w(\ell,\sigma',\sigma) \tag{35}$$

$$\overset{\lambda=1}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + x(\ell-1,\omega,\sigma')k(\ell,\sigma'',\sigma')^Tk(\ell,\sigma'',\sigma')w(\ell,\sigma',\sigma) \tag{36}$$

$$\overset{\text{Scaling}}{\Longrightarrow} \quad x(\ell) = x(\ell-1) + \mathrm{softmax}(x(\ell-1,\omega,\sigma')k(\ell,\sigma'',\sigma')^T)k(\ell,\sigma'',\sigma')w(\ell,\sigma',\sigma) \tag{37}$$

$$\overset{Q,K,V \text{ Notations}}{\Longleftrightarrow} \quad x(\ell) = x(\ell-1) + w(\ell)\,\mathrm{softmax}(QK^T)V \tag{38}$$

In the above derivation, Equation (34) assumes the Topology to be discrete, or the neighbourhood of a spatial point is itself, which restricts the convolution to be on a 1×1 window. For the matrix $w(\ell,\sigma,\sigma')$ with $\sigma,\sigma' \in [C]$, Equation (35) applies the linear transform k^Tk, where $k(\ell,\sigma'',\sigma)$ can be regarded as a set of C'' condition "pixels" of dimension C, or $\omega \in [C], \omega'' = [C'']$. Equation (36) assumes λ to be identity function denoted as 1. Equation (37) scales xw^T with a softmax function $\mathrm{softmax}(x(\sigma,\sigma'')) = \exp(x(\sigma,\sigma''))/\sum_{\sigma''\in[C'']}\exp(x(\sigma,\sigma''))$. Finally, Equation (37) is obtained from setting the Query-Key-Value notations $Q = x(\ell-1,\omega,\sigma'), K = V = k(\ell,\sigma',\sigma'')$. Note that by cancelling the assumption in Equation (26), we may also take in $Q = w_Q(\ell,\sigma,\sigma')Q', K = w_K(\ell,\sigma,\sigma')K', V = w_V(\ell,\sigma,\sigma')V'$.

Proposition G.5 (Attention Network Dynamics)**.**

$$\overset{\text{Attention Dynamics}}{\Longrightarrow} \quad \dot{x} = \mathrm{softmax}(QK^T)V \tag{39}$$

Equation (39) is obtained by setting $w(\ell)$ as identity and consider $\ell \in [0, L]$.

Proposition G.6 (ResNet Dynamics)**.** *By assuming the continuation $\ell \in [0, L]$, we obtain the continuous dynamics of ResNet*

$$(39) \overset{\text{ResNet Dynamics}}{\Longrightarrow} \quad \dot{x} = \lambda(x) \tag{40}$$