Affine Invariance in Continuous-Domain Convolutional Neural Networks

Anonymous authors
Paper under double-blind review

Abstract

The notion of group invariance helps neural networks in recognizing patterns and features under geometric transformations. Group convolutional neural networks enhance traditional convolutional neural networks by incorporating group-based geometric structures into their design. This research studies affine invariance on continuous-domain convolutional neural networks. Despite other research considering isometric invariance or similarity invariance, we focus on the full structure of affine transforms generated by the group of all invertible 2×2 real matrices (generalized linear group $\mathrm{GL}_2(\mathbb{R})$). We introduce a new criterion to assess the invariance of two signals under affine transformations. The input image is embedded into the affine Lie group $G_2 = \mathbb{R}^2 \ltimes \mathrm{GL}_2(\mathbb{R})$ to facilitate group convolution operations that respect affine invariance. Then, we analyze the convolution of embedded signals over G_2. In sum, our research could eventually extend the scope of geometrical transformations that usual deep-learning pipelines can handle.

1 Introduction

Traditional neural networks are very important in practice and have demonstrated strong performance across a wide range of tasks Abiodun et al. (2018); Widrow et al. (1994); Akbari et al. (2022); Shahin et al. (2001). However, they face computational challenges when dealing with high-dimensional inputs and lack built-in mechanisms for translation invariance Thompson et al. (2020); Mohaddes & Lederer (2025). Convolutional Neural Networks (CNNs) address these limitations by leveraging local connectivity and shared weights, making them well-suited for tasks like image recognition where spatial structure and translation invariance are essential O'Shea & Nash (2015). CNNs have achieved remarkable success in analyzing, recognizing, and understanding images, primarily due to their ability to automatically extract useful features from raw data. However, the types of structures that CNNs can capture are often limited to simple symmetries, which may restrict their effectiveness in more complex pattern recognition scenarios. In ordinary CNNs, translation symmetry of signals can often be detected due to the inherent translation equivariance of convolutional layers. In Group-Convolutional Neural Networks (G-CNNs), this idea is extended to more general group symmetries, enabling better handling of more complex transformations and potentially improving learning efficiency.

In this paper, we aim to develop a group convolutional neural network that categorizes two input signals as belonging to the same category, when these two signals can be converted into each other through an affine transformation. For example consider two input pictures, $\Phi_1 : \mathbb{R}^2 \to \mathbb{R}$ and $\Phi_2 : \mathbb{R}^2 \to \mathbb{R}$. An interesting question is whether there is any affine transform that can transform the first picture into the second. This question is akin to investigating if $\Phi_1(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) = \Phi_2(\boldsymbol{x})$, or at least if the distance between these two images is small. Selecting a proper kernel can lead to stability of convolution under affine transform (by stability under affine transform we mean $\Phi(\boldsymbol{x}) * g = \Phi(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) * g$).

However, as we will discuss later, it can be shown that Equation (2) does not have a solution in the general case and conventional convolutional neural network architectures are not useful in detecting general affine transforms. Therefore, we need to employ a different three layer convolutional neural network architecture. The idea is to embed the input image into G_2 (a process we call lifting), and perform convolutions in G_2,

which is invariant under affine transformation. This first layer is called the lifting layer. Then we need to perform the convolution in the convolution layer and finally, we have the projection layer which maps the signal in G_2 back to ordinary signals. Theorems 2, 3, and 4 show that if distance of an input function and a suitable affine transform of another input function is less than ϵ for a fixed $\boldsymbol{A}$ and $\boldsymbol{b}$, they are ϵ'-affine invariant in the lifting layer, convolutional layer, and projection layer, respectively. Therefore the output of the designed convolutional neural network is $c\epsilon$-affine invariant (for a constant c). We show that this network design is stable for affine invariant inputs. Furthermore, we demonstrate that the associated computations for these architectures can be simplified from complex convolutions over the transformation group to more straightforward integrals over Euclidean space.

This computational advantage becomes particularly meaningful in the context of Group Convolutional Neural Networks (G-CNNs), which extend CNNs to capitalize on the intrinsic geometric properties and symmetries in data, particularly images (Cohen & Welling, 2016). Unlike their traditional counterparts, G-CNNs harness the power of group theory, a mathematical framework that formalizes transformations and symmetries. This theoretical foundation ensures equivariance with respect to transformations described by the group, thereby enabling the network to maintain predictable behavior under various transformations.

One striking characteristic of G-CNNs is their ability to characterize geometric features and symmetries throughout the network's architecture. Notably, they excel when dealing with large groups that extend beyond mere translation equivariance. Classical CNNs can be regarded as a special instance of G-CNNs. The real power of G-CNNs becomes evident when more intricate geometric transformations are at play. Recent G-CNNs elevate feature maps to higher-dimensional, disentangled representations (Bekkers, 2019). Within these representations, G-CNNs effectively learn intrinsic geometric patterns and transformation behaviors present in the data—such as orientation, scale, and position—thereby reducing the need for traditional geometric data-augmentation techniques. This not only streamlines the learning process but may also reduces the risk of overfitting. Moreover, G-CNNs maintain their predictive behavior under geometric transformations, thanks to their foundation in group theory and, therefore, give rise to the concept of equivariance. The introduction of G-CNNs to the machine-learning community by (Cohen & Welling, 2016) marked the inception of an expanding body of G-CNN literature that consistently highlights many advantages of G-CNNs over conventional CNNs. This literature can be roughly classified into three main categories: discrete G-CNNs, regular continuous G-CNNs, and steerable continuous G-CNNs. Discrete G-CNNs delve into discrete group structures, yielding improved performance in various applications. This approach has been explored in studies by (Cohen & Welling, 2016; Winkels & Cohen, 2018; Dieleman et al., 2016; Worrall & Brostow, 2018; Hoogeboom et al., 2018), collectively contributing to the foundational understanding and practical deployment of discrete G-CNNs.

Regular continuous G-CNNs, as investigated by (Oyallon & Mallat, 2015; Bekkers et al., 2015; Weiler et al., 2018; Zhou et al., 2017), focus on seamless transformations within continuous domains. Their research showcases how G-CNNs can excel in handling continuous data, offering advantages over traditional CNNs in capturing intricate patterns and representations. Steerable continuous G-CNNs, explored (Cohen et al., 2018; Worrall et al., 2017; Kondor & Trivedi, 2018; Thomas et al., 2018; Andrearczyk et al., 2019), introduce a specialized approach where the convolution kernels are represented in terms of circular or spherical harmonics. This technique, particularly suitable for unimodular groups like roto-translations, enables efficient computation by utilizing basis coefficients. (Knigge et al., 2022) employs separability convolutions to attain equivariance concerning scale-rotation-translation transformations. (Chen et al., 2021) utilizes separability for efficient implementations of SE(3) equivariance.

Our research investigates the property of affine invariance in the context of continuous-domain convolutional neural networks. Our focus is on affine spaces formed by the generalized linear group $\mathrm{GL}_2(\mathbb{R})$, the group of all invertible matrices of size 2×2. Affine transformations are fundamental operations that combine linear transformations and translations. These transformations are important because they address distortions of an affine nature. For example, such distortions arise in certain types of CAPTCHA (Wang & Lu, 2018; Fisher et al., 2000; Guo et al., 2019) (see Fig. 1). Previous attempts have been made to investigate spaces that maintain affine-equivariance, but they are restricted to strict conditions, such as cases where the determinant equals 1 (expressed as $\mathrm{SO}(n)$). We instead consider affine-invariant spaces across the entire spectrum of invertible matrices. We introduce an approach, where we assess whether the convolution of the lifting of

Φ_1 and Φ_2 to G_2 exhibits G_2 invariance for every kernel. In order to apply this criterion, an additional step is required, namely, the computation of convolutions over G_2. We solve this technical challenge using QR-decomposition discussed in (Schindler, 1993).

This paper makes two main technical contributions:

1. We show that G-CNN architectures are stable under affine transformations generated by the generalized linear group $\mathrm{GL}_2(\mathbb{R})$. This includes a large spectrum of transformations, such as roto-translations and scale-translations. Thus, this paper is the first that proves the suitability of G-CNN architectures in such a generality.

2. We show that the related computations for these architectures can be simplified from complicated convolutions over the transformation group to much simpler integrals over the real space.

More broadly speaking, we show that G-CNNs cater to a considerably broader spectrum of transformations than what was established before. This is a general result, but it can also be used in specific scenarios to analyze invariant inputs in affinely generated transformations, such as the roto-translation transformation.

Figure 1: Original letters and their corresponding affine-invariant CAPTCHAs.

To facilitate a deeper exploration of G-CNNs, it is necessary to first review several fundamental definitions and concepts. We begin by revisiting several fundamental definitions and theoretical concepts. The following section introduces the essential framework, notation, and illustrative examples that form the foundation for the analyses and contributions presented in this paper, particularly in the context of investigating invariance properties within continuous-domain convolutional neural networks.

2 Preliminaries

Ensuring the equivariance of artificial neural networks (NNs) with respect to a group G is an essential characteristic, as it guarantees that applying transformations to the input preserves all information, merely shifting it to different network locations. It has been determined that when aiming for equivariant NNs, the sole viable choice is to employ layers in which the linear operator is defined through group convolutions. We begin by examining what a neural network framework for continuous signals looks like. For vector spaces, the most general form of a linear transformation is a matrix–vector multiplication, while for continuous signals, linear transformations are expressed through kernel operators. A neural network framework for handling continuous signals can be formulated using the following equation:

$$\boldsymbol{y} = \sigma\left(\mathcal{K}\Phi(\boldsymbol{x})\right), \tag{1}$$

where $\boldsymbol{x} \in \mathcal{X}$ represents the input vector, and $\Phi(\boldsymbol{x})$ denotes the input signal. For example in an image input, $\boldsymbol{x}$ is the location of pixels and $\Phi(\boldsymbol{x})$ is the value that image takes in each pixel. Moreover, $\mathcal{K} : \mathbb{L}_2(\mathcal{X}) \to \mathbb{L}_2(\mathcal{Y})$ denotes a linear map, and σ is the activation function. The kernel operator $\mathcal{K}$ is also defined as follows

$$\Phi^{l+1} = \mathcal{K}\Phi^l = \int_{\mathcal{X}} K(\boldsymbol{x}, \boldsymbol{y})\Phi^l(\boldsymbol{x})\mathrm{d}\lambda_{\mathcal{X}}(\boldsymbol{x}),$$

where $\mathcal{K} : \mathbb{L}_2(\mathcal{X}) \to \mathbb{L}_2(\mathcal{Y})$, $d\lambda_{\mathcal{X}}$ is a Radon measure on $\mathcal{X}$, $K(\boldsymbol{x}, \boldsymbol{y})$ denotes the kernel function, and $\Phi^l \in \mathbb{L}_2(\mathcal{X})$ denotes the feature map at layer l, i.e., a square-integrable function that serves as the input to the next layer in a continuous neural network. To broaden the application of this explanation to the notion of group convolutional neural networks, we revisit a number of crucial definitions.

Definition 1 (Group). *A group $(G, \cdot)$ is a set G equipped with a binary operator represented by a dot symbol. The dot operator is associative $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$, has an identity element e. Moreover, every element of the set has an inverse element $g \cdot g^{-1} = g^{-1} \cdot g = e$.*

In our considerations, the set comprises functions, such as translations or rotations. The group operation operates on elements of this set through addition or multiplication Herstein (1991). We also need to define normal groups. A normal group is a subgroup N of a group G such that, for every element g in G, the conjugate gNg^{-1} is contained within N.

Example 1 (Translation group). *The translation group in $\mathbb{R}^2$ is denoted by $(\mathbb{R}^2, \cdot)$ consists of all possible translations and is equipped with the below group product and group inverse:*

$$g \cdot g' = (\boldsymbol{x} + \boldsymbol{x}')$$
$$g^{-1} = -\boldsymbol{x},$$

where $g = (\boldsymbol{x})$ and $g^{-1} = (-\boldsymbol{x})$ and $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^2$.

One important example of groups are Lie groups, which are defined as follows:

Definition 2 (Lie groups). *A Lie group is a set G with two structures: G is a group and G is a (smooth, real) manifold. These structures agree in the following sense: multiplication and inversion are smooth maps.*

Roto-translation symmetries of Euclidean spaces are examples of Lie groups, which is explained in the next example.

Example 2 (Roto-translation group). *The roto-translation group in $\mathbb{R}^2$ is denoted by $\mathrm{SE}(2)$. The group $\mathrm{SE}(2) = \mathbb{R}^2 \rtimes \mathrm{SO}(2)$ consists of translations vectors in $\mathbb{R}^2$, and rotations in $\mathrm{SO}(2)$ and is equipped with the group product and group inverse:*

$$g.g' = (\boldsymbol{x}, \boldsymbol{R}_\theta) \cdot (\boldsymbol{x}', \boldsymbol{R}_{\theta'}) = (\boldsymbol{R}_\theta \boldsymbol{x}' + \boldsymbol{x}, \boldsymbol{R}_{\theta+\theta'})$$
$$g^{-1} = (-\boldsymbol{R}_\theta^{-1}\boldsymbol{x}, \boldsymbol{R}_\theta^{-1}),$$

for $g = (\boldsymbol{x}, \boldsymbol{R}_\theta)$, $g' = (\boldsymbol{x}', \boldsymbol{R}_{\theta'})$, $\boldsymbol{R}_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, and identity element $(\boldsymbol{0}, \boldsymbol{I})$. Note that $\rtimes$ denotes the semidirect product. In a direct product $G = H \times K$, both H and K are normal in G. Semidirect products are a relaxation of direct products where only one of the two subgroups must be normal.

The group operator provides instructions on how to act on the group elements, ensuring that the result remains within the group. Of particular interest are symmetry groups, where each element in the set represents a symmetry transformation. When the group acts on a specific space, it is referred to as a group action.

Definition 3 (Group action). *Let χ be a set. If G is a group with identity element e, then a group action α of G on χ is a function, $\alpha : G \times \chi \to \chi$, (which is usually denoted as $\alpha(h, \boldsymbol{x}) = h \odot \boldsymbol{x}$) that satisfies identity and compatibility conditions ($e \odot \boldsymbol{x} = \boldsymbol{x}$, $g \odot (h \odot \boldsymbol{x}) = (g \cdot h) \odot \boldsymbol{x}$) for all $g, h \in G$ and all $\boldsymbol{x} \in \chi$.*

For example the action of group $G = \mathrm{SO}(d)$ on space $\chi = \mathbb{R}^d$ could be denoted by $g \odot \boldsymbol{x} = \boldsymbol{R}\boldsymbol{x}$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{R} \in \mathrm{SO}(d)$. For the set of points, we perform transformation through group products, while in the convolution kernel, we perform transformation via group representations. Therefore we need to understand representations. The multiplication within a group instructs us on merging transformations, yet it does not provide guidance on utilizing these transformations on other entities like vectors or signals. To address this, we require the concept of group action and group representations. Nevertheless, frequently, our attention is predominantly directed towards linear group actions operating on vector spaces, and these actions are termed representations.

Definition 4 (Representation). *A representation is an invertible linear transformation $\rho(g) : V \to V$ parameterized by a group elements $g \in G$ that acts on some vector space V, which follows the group structure (it is a group homomorphism) via*

$$\rho(g)\rho(h)v = \rho(g \cdot h)v$$

for $v \in V$ and $g, h \in G$.

A standard approach for defining the action of a group G on functions is via the regular representation.

Definition 5 (Regular representation)**.** *Let $\Phi \in \mathbb{L}_2(\mathcal{X})$. Then the regular representation of G acting on $\mathbb{L}_2(\mathcal{X})$ is given by*

$$\rho(g)\Phi(\boldsymbol{x}) = \Phi\left(g^{-1}\boldsymbol{x}\right).$$

Example 3 (Regular representation of roto-translation group)**.** *Let $\Phi \in \mathbb{L}_2(\mathbb{R}^2)$ be a two dimensional image, $G = \mathrm{SE}(2)$ denotes the roto-translation group then*

$$\rho(g)\Phi(\boldsymbol{y}) = \Phi(\boldsymbol{R}_\theta^{-1}(\boldsymbol{y} - \boldsymbol{x})).$$

We continue this part with some additional definitions needed in the next part.

Definition 6 (Coset)**.** *Let $H \subset G$ be a subgroup of G. Then gH denotes a coset given by*

$$gH = \big\{g \cdot h \mid h \in H\big\}.$$

Definition 7 (Quotient Space)**.** *Let $H \subset G$ be a subgroup of G. Then G/H denotes the quotient space that is defined as the collection of unique cosets $gH \subset G$. Elements of G/H are thus cosets that represents an equivalence class of transformations for which $g \sim \tilde{g}$ are equivalent if there exists an $h \in H$ such that $g = \tilde{g}h$.*

Definition 8 (Stabilizer)**.** *Let G act on $\mathcal{X}$ via the action $\odot$. For every $\boldsymbol{x} \in \mathcal{X}$, the stabilizer subgroup of G with respect to the point $\boldsymbol{x}$, denoted as $\mathrm{Stab}_G(\boldsymbol{x})$, is the set of all elements in G that fix $\boldsymbol{x}$, i.e.*

$$\mathrm{Stab}_G(\boldsymbol{x}) = \big\{g \in G \mid g \odot \boldsymbol{x} = \boldsymbol{x}\big\}.$$

Moreover from Bekkers (2019) we know that, if $\mathcal{X}$ is a homogeneous space (informally, a homogeneous space can be understood as a space that possesses a uniform structure throughout, in the sense that every point can be transformed into any other by the action of a group.) of G, then $\mathcal{X}$ can be identified with G/H with $H = \mathrm{Stab}_G(\boldsymbol{x}_0)$ for any $\boldsymbol{x}_0 \in \mathcal{X}$. For simplicity in notation we do not use $\cdot$ and $\odot$ symbols in the next sections. Furthermore, in this paper we use g and h to denote group elements and Φ and K to denote input and kernel functions respectively.

3 Group Convolutional Neural Networks Architecture

One conventional method to build group convolutional neural networks is to apply isotropic convolutions for Equation (1). An isotropic $\mathbb{R}^d$ convolutional layer maps between planar signals $\mathbb{L}_2\left(\mathbb{R}^d\right)$ with $\mathcal{K}$ a planar correlation given by

$$(\mathcal{K}\Phi)(\boldsymbol{y}) = \int_{\mathbb{R}^d} K(\boldsymbol{x} - \boldsymbol{y})\Phi(\boldsymbol{x})\mathrm{d}\boldsymbol{x},$$

and in which it is shown (Theorem 1 from Bekkers (2019)) that if $\mathcal{Y} \equiv G/H$ is the quotient of G with $H = \mathrm{Stab}_G(y_0) = \{g \in G \mid gy_0 = y_0\}$, then the kernel K satisfies

$$\text{for all } h \in H: \quad K(\boldsymbol{x}) = \frac{1}{|\det h|} K\left(h^{-1}\boldsymbol{x}\right). \tag{2}$$

Applying isotropic convolutions is limiting because they are constrained by the shape of the kernels. One approach to overcome this limitation, is to lift the signals to a group G. Lifting of the input signal, not only addresses the constraints of kernels as noted by Bekkers (2019) but also offers advantages in enhancing the performance of image processing, as highlighted in the work by Smets et al. (2023). When we apply lifting we must look for stabilizer Stab_G when G acts on G. In this case we have

$$H = \mathrm{Stab}_G(g) = \big\{x \in G | xg = g\big\} = \{e\}.$$

As a result, Equation (2) is fulfilled for all kernels, and there are no longer any limitations imposed on the choice of kernels.

Definition 9 (Lifting layer ($\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = G_2$))**.** *Let $g = (\boldsymbol{x}, \boldsymbol{P}) \in G_2$, where $\boldsymbol{x} \in \mathbb{R}^2$ and $\boldsymbol{P} \in \mathrm{GL}_2(\mathbb{R})$. A lifting layer maps functions from $\mathbb{L}_2\left(\mathbb{R}^d\right)$ to $\mathbb{L}_2(G_2)$, where G_2 is a group, via a lifting correlation defined by:*

$$(\mathcal{K}f)(g) = \int_{\mathbb{R}^d} \frac{1}{|\det \boldsymbol{P}|} K\left(g^{-1}\tilde{\boldsymbol{x}}\right) \Phi(\tilde{\boldsymbol{x}}) \mathrm{d}\tilde{\boldsymbol{x}}.$$

Example 4 (Lifting for Kronecker delta kernel)**.** *Let*

$$K = \delta(\boldsymbol{x}, \mathbf{0}_{d\times d}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \mathbf{0} \in \mathbb{R}^d; \\ 0 & \text{otherwise.} \end{cases}$$

Then for $g = (\boldsymbol{x}, \boldsymbol{P})$ we have

$$g^{-1}\tilde{\boldsymbol{x}} = \boldsymbol{P}^{-1}(\tilde{\boldsymbol{x}} - \boldsymbol{x}).$$

Therefore,

$$K(g^{-1}\tilde{\boldsymbol{x}}) = \delta(\boldsymbol{P}^{-1}(\tilde{\boldsymbol{x}} - \boldsymbol{x}), \mathbf{0}_{d\times d}) = \begin{cases} 1 & \text{if } \tilde{\boldsymbol{x}} = \boldsymbol{x}; \\ 0 & \text{otherwise.} \end{cases}$$

This implies that the lifting layer is given by

$$(\mathcal{K}\Phi)(g) = \frac{\Phi(\boldsymbol{x})}{|\det \boldsymbol{P}|}.$$

We also need to discuss the existence of the lifting layer integral. A function Φ on $\mathbb{R}$ is called locally integrable if Φ is integrable on every bounded interval $[a, b]$ for $a < b$ in $\mathbb{R}$. If $K \in \mathcal{C}_c^\infty(\mathbb{R})$ and Φ is locally integrable, then

$$(\Phi * K)(y) = \int_{-\infty}^{\infty} \Phi(t) K(y - t) \mathrm{d}t,$$

exists and is infinitely differentiable on $\mathbb{R}$. First of all the input Φ is usually a picture and therefore the function Φ is bounded. On the other hand the value of lifted functions on cosets is equal to that of Φ. Therefore the lifted function is bounded as well. We further know that the kernel is locally supported, which results the integrability. After lifting layer we will apply convolutional layer which is defined as follows.

Definition 10 (Group convolutional layer ($\mathcal{X} = \mathcal{Y} = G$))**.** *A group convolutional layer maps between G-feature maps in $\mathbb{L}_2(G)$. A group convolution is given by*

$$(\Phi * K)(h) = \int_G \Phi(h) K\left(h^{-1}g\right) \mathrm{d}\mu_G,$$

where $g \in G$ and μ_G is a Haar measure.

We also need another layer to again maps to feature maps in $\mathbb{L}_2(\mathbb{R}^d)$. Let $G = \mathbb{R}^d \rtimes \tilde{H}$. Then we have the following definition:

Definition 11 ($\mathbb{R}^d$ Projection layer)**.** *A projection layer maps between G-feature maps in $\mathbb{L}_2(G)$ back to planar feature maps in $\mathbb{L}_2(\mathbb{R}^d)$*

$$(\mathcal{K}\Phi)(\boldsymbol{x}) = \int_{\tilde{H}} \Phi(\boldsymbol{x}, \tilde{h}) \mathrm{d}\tilde{h}. \tag{3}$$

Frequently, the focus is on constructing architectures that are invariant, as opposed to equivariant. Invariance with respect to all transformations in G is accomplished through mean pooling across the entire group G, akin to how global translation invariance is typically obtained by mean or max pooling over the spatial dimensions of feature maps. The global pooling layer can be defined as follows.

Definition 12 (Global pooling layer)**.** *A global pooling layer transforms any feature map into a single scalar value as follows,*

$$(\mathcal{K}\Phi) = \int_{\mathcal{X}} \Phi(x) \mathrm{d}\mu(x), \tag{4}$$

where $\mathcal{K}$ represents a pooling operation over $\mathcal{X}$ and $d\mu(x)$ is a Radon measure on $\mathcal{X}$.

4 Principal Results

Here, we provide the principal result established in this section. In particular we explore affine invariant spaces and investigate the convolution integration over G_2.

4.1 Problem Statement

Our goal is to study invariance in affine transformations in continuous-domain convolutional neural networks. An affine transformation basically combines linear transformations and translations. Affine transformations are denoted as follows

$$G_2 = \Big\{[\boldsymbol{x}, \boldsymbol{A}] : \boldsymbol{x} \in \mathbb{R}^2, \boldsymbol{A} \in \mathrm{GL}_2(\mathbb{R})\Big\},$$

where

$$[\boldsymbol{x}, \boldsymbol{A}] : \boldsymbol{z} \mapsto \boldsymbol{x} + \boldsymbol{A}\boldsymbol{z}.$$

The identity is $[\boldsymbol{0}, \boldsymbol{I}]$, and, therefore, for all $\boldsymbol{B} \in \mathrm{GL}_2(\mathbb{R})$ we have $[\boldsymbol{y}, \boldsymbol{B}]^{-1} = [-\boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{B}^{-1}]$.

The affine transformation is important as we may face affine type distortions due to varying proximity of the camera with respect to the object. For example, this type of affine distortion could manifest in remote sensing images, as well as in camera imagery which can include various perspective distortions Fisher et al. (2000). It is important to note that in an affine transformation, parallel lines in the original image continue to remain parallel in the transformed image. However, the transformation can introduce distortion in the angles between lines. In order to study resemblance of two input functions Φ_1 and Φ_2 we need to provide some more definitions.

Definition 13 (ϵ-equivalance)**.** *We say that functions $\Phi_1, \Phi_2 \in \mathbb{L}(\mathbb{R}^2)$ are ϵ-equivalent if $\|\Phi_1 - \Phi_2\|_1 < \epsilon$ or $\sup_{\boldsymbol{x}} |\Phi_1(\boldsymbol{x}) - \Phi_2(\boldsymbol{x})| < \epsilon$.*

For instance, if a picture shows a slight deviation due to noise, we aim to overlook or disregard this deviation.

Definition 14 (Affine invariance)**.** *We say that functions $\Phi_1, \Phi_2 \in \mathbb{L}(\mathbb{R}^2)$ are Affine invariant if there exists $h \in G_2$ so that $\Phi_1 = \rho(h)\Phi_2$.*

Definition 15 (ϵ-Affine invariance)**.** *We say that functions $\Phi_1, \Phi_2 \in \mathbb{L}(\mathbb{R}^2)$ are ϵ-Affine invariant if there exists $h \in G_2$ so that $\|\Phi_1 - \rho(h)\Phi_2\|_1 < \epsilon$ or $\sup_{\boldsymbol{x}} |\Phi_1(\boldsymbol{x}) - \rho(h)\Phi_2(\boldsymbol{x})| < \epsilon$.*

This paper explores the use of convolutional neural networks in handling affine transformations, focusing specifically on cases where the transformation matrix $\boldsymbol{A}$ belongs to the general linear group $\mathrm{GL}_2(\mathbb{R})$. We diverge from the use of isometric convolutions, opting instead for the application of the lifting-projection method, which we have elucidated in Section 3 comprehensively. While prior investigations have focused on compact groups such as SO(2), it is important to highlight that the $\mathrm{GL}_2(\mathbb{R})$ group does not fall under the category of compact groups. Our alternative method focuses on analyzing the convolution of the lifted forms of the signals Φ_1 and Φ_2 for achieving G_2 invariance. We also need to introduce an extra step that encompasses performing convolutions on G_2 and address the unique challenges associated with this, including techniques for handling integrations over G_2. Therefore, the initial step is to demonstrate that this three-layer group-convolutional neural networks are stable under affine transformations produced by the generalized linear group $\mathrm{GL}_2(\mathbb{R})$. The below theorem addresses this.

Theorem 1 (Stability of G-CNN architecture)**.** *Let Σ be the G-CNN consisting of three, lifting, convolutional, and $\mathbb{R}$-projection layer and let the distance of a function Φ_1 and the affine transform of another function Φ_2 be less than ϵ, then*

$$|\Sigma\Phi_1 - \Sigma\Phi_2| < c\epsilon,$$

where $c = \|K_1\|_1^{\mathbb{R}^2} \|K_2\|_1^{G_2}$ and K_1 and K_2 are the kernels of the lifting layer and the convolutional layer, respectively.

Proof. We analyze the three layers of the network individually, demonstrating the stability of each layer. This, in turn, ensures the stability of the entire network. Theorems 2, 3, and 4 detail this process. The current theorem is a direct consequence of these theorems. □

Remark 1. *Example 4 implies that the stability of affine-invariant systems can decrease as the determinant of the transformation matrix approaches zero. This aligns with geometric intuition: as* $\det(\boldsymbol{P}) \to 0$*, the transformation projects initial objects up to a very small distance to lower-dimensional subspace. Consequently, inversion or reconstruction becomes more challenging.*

The next theorem asserts that, the lifting layer does not change the affine invariance of input signals. The proof of the following theorems can be found in Section A.

Theorem 2 (Invariance in the lifting layer)**.** *Let* $\Phi_1, \Phi_2 : \mathbb{R}^2 \to \mathbb{R}$ *be input signals and let there exists an* $g \in G_2$ *so that* $\sup_g |(\Phi_1 - \rho(g^{-1})\Phi_2| < \epsilon$ *then*

$$\sup_g |(\mathcal{K}\Phi_1)(g) - \rho(g^{-1})(\mathcal{K}\Phi_2)(g)| < \epsilon \|K\|_1^{\mathbb{R}^2}.$$

The next step is to demonstrate the invariance in the convolutional layer.

Theorem 3 (Invariance in the convolutional layer)**.** *Let* $(\mathcal{K}\Phi_1), (\mathcal{K}\Phi_2) : G_2 \to \mathbb{R}$ *be the lifting of* $\Phi_1, \Phi_2 : \mathbb{R}^2 \to \mathbb{R}$*. If there exists an* $\tilde{h} \in G_2$ *so that* $\|(\mathcal{K}\Phi_1)(g) - \rho(\tilde{h})(\mathcal{K}\Phi_2)(g)\|_{\sup} < \epsilon$ *then*

$$\|(\mathcal{K}\Phi_1) * K - \rho(\tilde{h})(\mathcal{K}\Phi_2) * K\|_{\sup}^{G_2} < \epsilon \|K\|_1^{G_2}$$

holds for every kernel K *and vice-versa. Where* $\|\Phi\|_1^{G_2} = \int_{G_2} |\Phi| \mathrm{d}\mu_{G_2}$*.*

The aforementioned finding indicates that to assess the equivalence of two signals, it is necessary to perform a convolutional integration across G_2. We investigate this problem in the next section. In the last step we provide the below theorem which states that the function defined by $\int_{G_2} (\mathcal{K}\Phi) * K \, \mathrm{d}\mu_{G_2}(g)$ which is a continuous function from $\mathcal{C}(\mathbb{R}^2, \mathbb{R})$ to $\mathbb{R}$ can be used for characterization of invariant affine functions in the projection and pooling layer.

Theorem 4 (Invariance in the projection and pooling layer)**.** *If* $(\mathcal{K}\Phi_1), (\mathcal{K}\Phi_2) : G_2 \to \mathbb{R}$ *are lifting of input signals and there exists a* $\tilde{h} \in G_2$ *such that* $\|(\mathcal{K}\Phi_1) - \rho(\tilde{h})(\mathcal{K}\Phi_2)\|_1^{G_2} < \epsilon$*. Then we have*

$$\Big| \int_{G_2} \big((\mathcal{K}\Phi_1) * K - (\mathcal{K}\Phi_2) * K\big)(h) \mathrm{d}\mu_{G_2}(h) \Big| \leq \epsilon \|K\|_1^{G_2}$$

4.2 Convolution Computation

Before illustrating how to compute the convolution over the group G_2, we remark some ingredients which are essential to compute the convolution over G_2. We finally show that the convolution over G_2 can be computed through Fourier transform and integration over real valued space.

In our study, we adopt a straightforward approach to calculate the G_2-invariant convolution for a kernel, which can be formulated as follows:

$$\int_{G_2} \Phi([\boldsymbol{x}, \boldsymbol{A}]) K([\boldsymbol{y}, \boldsymbol{B}]^{-1}[\boldsymbol{x}, \boldsymbol{A}]) \mathrm{d}\mu_{G_2}. \tag{5}$$

Using the Stone–Weierstrass theorem, in the setup of continuous functions with respect to sup-norm, $\mathcal{C}(G_2, \mathbb{R}) = \mathcal{C}(\mathrm{GL}_2(\mathbb{R}) \ltimes \mathbb{R}^2, \mathbb{R})$, which asserts that sums of separable functions are dense in $\mathcal{C}(G_2, \mathbb{R})$, we reduce the kernel sets to functions of the form $K(\boldsymbol{y}, \boldsymbol{A}) = \sum_{i=1}^{M} K_{1_i}(\boldsymbol{y}) K_{2_i}(\boldsymbol{A})$. This reduction helps us to benefit from Fourier transforms to simplify some parts of our calculations. We use QR parametrization of $\mathrm{GL}_2(\mathbb{R})$ which aids us in utilizing numerical approaches, for example those introduced in Eshkuvatov et al. (2013). Now, we illustrate the outcomes presented in Milad & Taylor (2023); Schindler (1993), which are pertinent to our calculations. Let

$$K_0 = \left\{ \begin{pmatrix} s & -t \\ t & s \end{pmatrix} : s, t \in \mathbb{R}, s^2 + t^2 > 0 \right\},$$

and

$$H_{(1,0)} = \left\{ \begin{pmatrix} 1 & 0 \\ u & v \end{pmatrix} : u, v \in \mathbb{R}, v \neq 0 \right\}.$$

It is shown that $\mathrm{GL}_2(\mathbb{R}) = K_0 H_{(1,0)}, K_0 \cap H_{(1,0)} = \boldsymbol{I}$, where $\boldsymbol{I}$ denotes the identity matrix, and $(\boldsymbol{M}, \boldsymbol{C}) \to \boldsymbol{MC}$ is a homeomorphism between $K_0 \times H_{(1,0)}$ and $\mathrm{GL}_2(\mathbb{R})$. From Milad & Taylor (2023) we have

$$\int_{G_n} \Phi \, \mathrm{d}\mu_{G_n} = \int_{\mathrm{GL}_n(\mathbb{R})} \int_{\mathbb{R}^n} \Phi[\boldsymbol{x}, \boldsymbol{A}] \frac{\mathrm{d}\boldsymbol{x} \mathrm{d}\mu_{\mathrm{GL}_n(\mathbb{R})}(\boldsymbol{A})}{|\det(\boldsymbol{A})|}, \text{ for all } \Phi \in C_c\left(G_n\right), \tag{6}$$

where $\mathcal{C}_c(G_n)$ denotes the space of continuous $\mathbb{R}$-valued functions of compact support on G_n. For any integrable function Φ on $\mathrm{GL}_2(\mathbb{R})$, the Haar integral on $\mathrm{GL}_2(\mathbb{R})$ can be expressed as

$$\int_{\mathrm{GL}_2(\mathbb{R})} \Phi \, \mathrm{d}\mu_{\mathrm{GL}_2(\mathbb{R})} = \int_{K_0} \int_{H_{(1,0)}} \Phi(\boldsymbol{MC}) |\det(\boldsymbol{C})| \mathrm{d}\mu_{H_{(1,0)}} \mathrm{d}\mu_{K_0}. \tag{7}$$

Let $\mathbb{R}^* = \mathbb{R} \setminus \{0\}$. Then the map

$$[u, v] \to \begin{pmatrix} 1 & 0 \\ u & v \end{pmatrix}$$

is an isomorphism between the group G_1 and $H_{(1,0)}$. When $n = 1, \mathrm{GL}_1(\mathbb{R})$ can be identified with $\mathbb{R}^*$ and G_1 can be identified with $\mathbb{R} \rtimes \mathbb{R}^*$. We recall that $\int_{\mathbb{R}^*} \Phi \mathrm{d}\mu_{\mathbb{R}^*} = \int_{\mathbb{R}} \Phi(b) \frac{\mathrm{d}b}{|b|}$, where the integral on the right hand side is the Lebesgue integral on $\mathbb{R}$, and by 6

$$\int_{G_1} \Phi \, \mathrm{d}\mu_{G_1} = \int_{\mathbb{R}} \int_{\mathbb{R}} \Phi[y, b] \frac{\mathrm{d}y \mathrm{d}b}{b^2}. \tag{8}$$

4.3 Integral over G_2

A difficult aspect in the implementation of group convolutional neural networks involves performing convolutions across the group. This segment addresses this particular challenge by delving into the problem, which we will break down into the more manageable tasks of calculating Fourier transforms and conducting integrations in real-valued space. We have the below theorem for the integration over G_2.

Theorem 5. *Let*

$$\boldsymbol{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{GL}_2(\mathbb{R})$$

and let the kernel be separable meaning that $K(\boldsymbol{x}, \boldsymbol{A}) = K_1(\boldsymbol{x}) K_2(\boldsymbol{A})$ and consider the one to one transform between H and H^* so that

$$H^*(s, t, u, v, \boldsymbol{B}, \boldsymbol{y}) := H_{\Phi,K}(a, b, c, d, \boldsymbol{B}, \boldsymbol{y}) = \frac{K_2(\boldsymbol{A}\boldsymbol{B}^{-1})}{|\det(\boldsymbol{A})||\det(\boldsymbol{B}^{-1})|} \mathcal{F}^{-1}\Big(\widehat{\Phi}(\boldsymbol{u}) \widehat{K}_1(\boldsymbol{B}^\top \boldsymbol{u})\Big),$$

where $a = s - ut$, $c = t + us$, $b = -t/v$, and $d = s/v$, then we have

$$\int_{G_2} \Phi([\boldsymbol{x}, \boldsymbol{A}]) K([\boldsymbol{y}, \boldsymbol{B}]^{-1}[\boldsymbol{x}, \boldsymbol{A}]) \mathrm{d}\mu_{G_2} = \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} H^*(s, t, u, v, \boldsymbol{B}, \boldsymbol{y}) \frac{\mathrm{d}u \mathrm{d}v}{|v|} \frac{\mathrm{d}s \mathrm{d}t}{s^2 + t^2}.$$

where $\mathcal{F}(K_1) = \widehat{K}_1(\boldsymbol{u})$ and $\mathcal{F}(\Phi) = \widehat{\Phi}(\boldsymbol{u})$ are Fourier transforms.

The proof of this theorem is discussed in Section A. Applying this result we can use the numerical methods in Eshkuvatov et al. (2013) to compute the former integral as it has singularity in $s = 0, t = 0$. Note that we can write K_0 as $\mathbb{R}^+ \rtimes \mathrm{SO}(1)$ where

$$\begin{pmatrix} s & -t \\ t & s \end{pmatrix} = (s^2 + t^2) \times \begin{pmatrix} r\cos\theta & -r\sin\theta \\ r\sin\theta & r\cos\theta \end{pmatrix}.$$

The final step that necessitates computation is the integration within the projection layer. In the context of our affine transformation, the stabilizer is specifically $\mathrm{GL}_2(\mathbb{R})$. We refrain from delving into the intricacies of this process, as it bears resemblance to the earlier scenario.

Now we first provide one example on G_1 this example gives us some insights for examples on G_2. We also will see how two class of complicated functions are equivalent with some intervals on $\mathbb{R}$ which are easier to investigate.

Example 5. *Let the input function be defined as*

$$\Phi(a) = \begin{cases} 1 & \text{if } a \in [t_1, t_2] \\ 0 & \text{otherwise} \end{cases},$$

then according to Example (4) for the lifting of Φ we have

$$(\mathcal{K}\Phi)[a,b] = \begin{cases} \frac{1}{|b|} & \text{if } a \in [t_1, t_2] \\ 0 & \text{otherwise} \end{cases},$$

then we have:

$$\begin{aligned} &\int_{\mathbb{R}^2} \mathcal{K}(\Phi[a,b]) K([x,y]^{-1}[a,b]) \mathrm{d}\mu_{G_1}(a,b) \\ &= \int_{\mathbb{R}^2} \frac{1}{|b|} K([-\frac{y}{x}, \frac{1}{x}][a,b]) \frac{\mathrm{d}a\mathrm{d}b}{b^2} \\ &= \int_{\mathbb{R}^2} \frac{1}{|b|b^2} K([\frac{a-y}{x}, \frac{b}{x}]) \mathrm{d}a\mathrm{d}b \end{aligned}$$

We can define a separable kernel so that we make the computations easier. We define it as $K(s,t) = s^3 \exp(s) \frac{1}{\sqrt{2\pi}} exp(-t^2)$. Employing this definition and assuming b is positive, we obtain:

$$\begin{aligned} &\int_{\mathbb{R}^2} \frac{1}{|b|b^2} K([\frac{a-y}{x}, \frac{b}{x}]) \mathrm{d}a\mathrm{d}b = \int_{t_1}^{t_2} \exp(\frac{a-y}{x}) \int_{\mathbb{R}} \frac{1}{|b|b^2} \frac{b^3}{x^3} \frac{1}{\sqrt{2\pi}} \exp -(\frac{b}{x})^2 \mathrm{d}a\mathrm{d}b \\ &= \int_{t_1}^{t_2} \exp(\frac{a-y}{x}) \int_{\mathbb{R}} \frac{1}{|b|b^2} \frac{b^3}{x^3} \frac{1}{\sqrt{2\pi}} \exp -(\frac{b}{x})^2 \mathrm{d}a\mathrm{d}b = \frac{1}{\sqrt{2}x} \exp(\frac{t_2-y}{x}) - \frac{1}{\sqrt{2}x} \exp(\frac{t_1-y}{x}) \end{aligned}$$

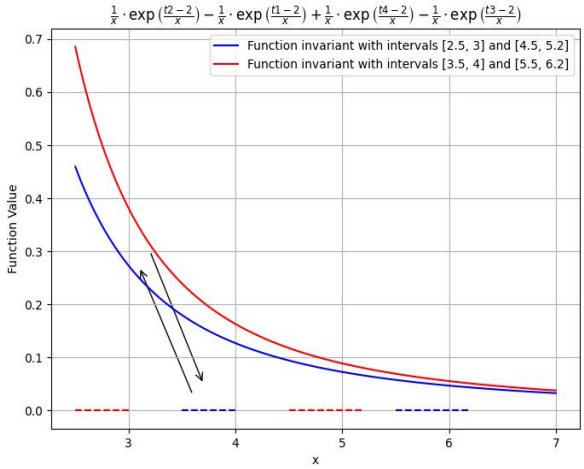


Figure 2: Two families of related affine invariant functions denoted by dashed and solid lines.

This example illustrates how invariance under affine transformations for more complex functions can be linked to the invariance of certain simple step functions. It also underscores the importance of selecting an appropriate kernel function. In the figure below, we selected two distinct intervals for input functions and then applied an affine transform [1,1]. It shows that we can study invariance under affine transformations of more complex shapes through the invariance of certain simple intervals.

5 Experiments

In this section, we present simulation results to demonstrate the practical significance of our theoretical framework. Specifically, we construct a neural network consisting of a three-layer group convolution module, followed by a fully connected neural network. The design of the group convolutional layers was guided by the methodologies described in Examples 4 and 5, which were used to define the kernels for the lifting and convolution operations. Projection is accomplished using straightforward averaging. For comparison, we also implemented a standard convolutional neural network that shares an identical fully connected component with the same hyperparameters. Both models were evaluated on a dataset generated using affine transformations to test their performance under geometric variability. As the results will show, our proposed group convolution network can outperform the conventional CNN, particularly in scenarios where the number of input samples is relatively low. In the following, we provide a more detailed explanation of our simulation setup and results.

To evaluate the proposed G-CNN, we generated a synthetic digit dataset using affine-transformed images of digits 0 through 9. Each digit was rendered using a standard font on a fixed-size grayscale canvas, followed by an affine transformation incorporating stretch, shear, rotation, and translation. A fixed number (N) of transformed samples were created per digit class, with translation parameters sampled uniformly to introduce spatial variability. The final dataset contains $10N$ labeled images, scalable via N, allowing for controlled experiments across models such as G-CNNs and CNNs.

In the first experiment, we used the following matrix to define our affine transformation

$$A_1 = \begin{bmatrix} 2.5 & 0.7 \\ 0.6 & 1.8 \end{bmatrix}.$$

As shown in Fig. 3, the G-CNN outperforms the standard CNN across all dataset sizes. The G-CNN demonstrates strong generalization even with limited data, achieving high accuracy with a few samples per class. In contrast, the standard CNN requires more data to reach comparable performance. We also trained our network on data generated using the matrix

$$A_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}.$$

As illustrated in Fig. 4, the G-CNN outperform CNN when the input data size is limited.

Furthermore, to better visualize and evaluate the performance of GCCNs relative to standard convolutional neural networks (CNNs), we conducted additional experiments using the following transformation matrices (as the linear parts of the affine maps):

$$A_1 = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}.$$

The evaluation uses a synthetic dataset with 40 training samples per digit. Each digit is subjected to an affine transformation whose linear component is one of the matrices above, combined with rotation, scaling (stretch), and translation, introducing substantial geometric variability. Figures 5, 6, and 7 present predictions for both models on a test set, with each sample annotated by the true label (T) and the predicted label (P). The G-CCN exhibits stronger affine invariance, correctly identifying more transformed digits than the standard CNN. These results underscore the benefit of incorporating group-equivariant structure in network design, particularly when significant geometric variability is present.

6 Conclusion

In this work, we have demonstrated that Group Convolutional Neural Networks (G-CNNs) exhibit stability under affine transformations arising from the full general linear group $\mathrm{GL}_2(\mathbb{R})$. This includes a wide range

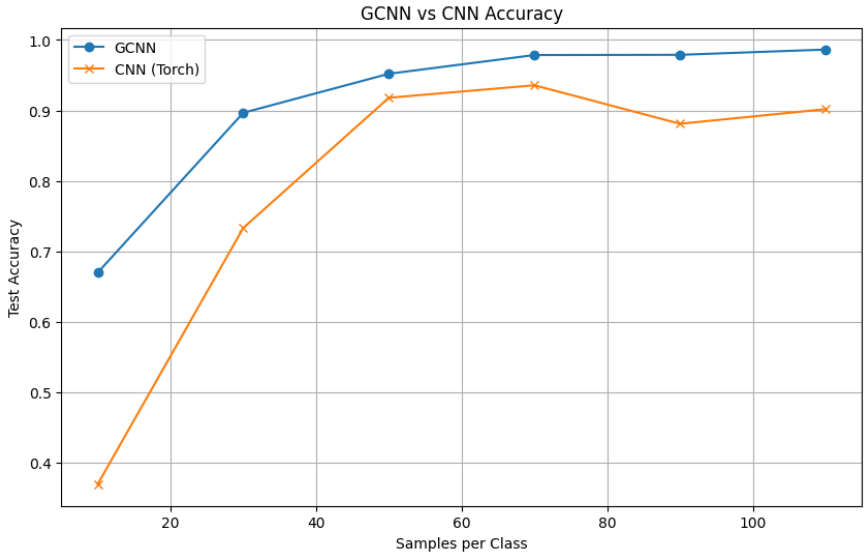


Figure 3: Accuracy comparison between G-CNN and standard CNN across varying sample sizes. G-CNN outperforms CNN under affine transformation $A = \begin{bmatrix} 2.5 & 0.7 \\ 0.6 & 1.8 \end{bmatrix}$.

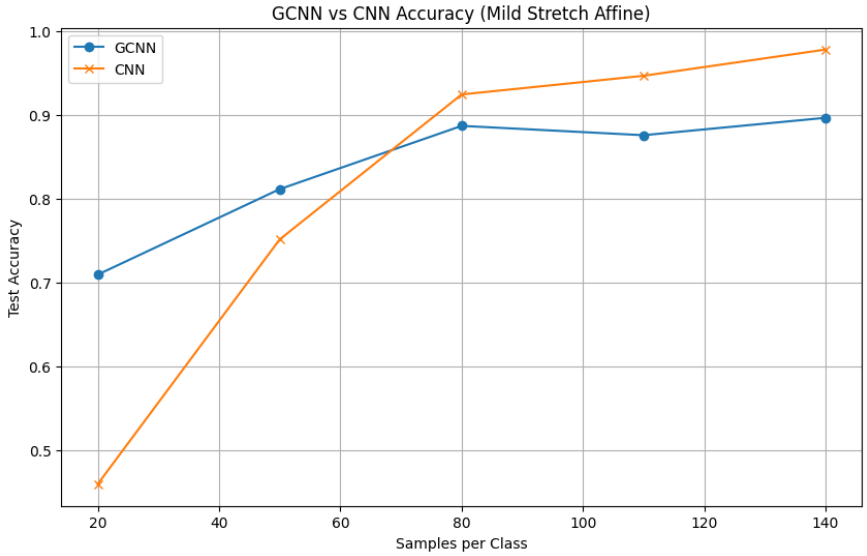
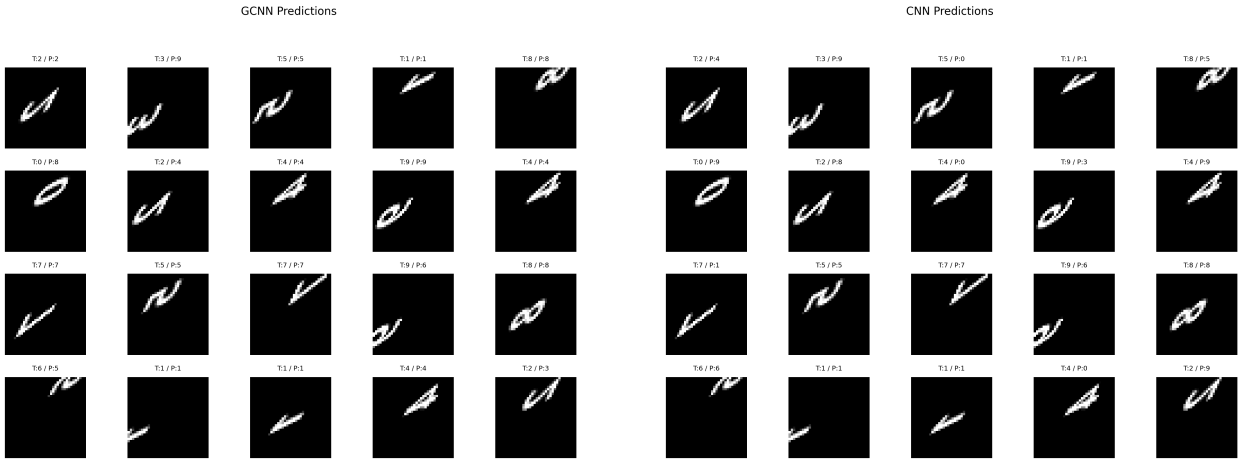
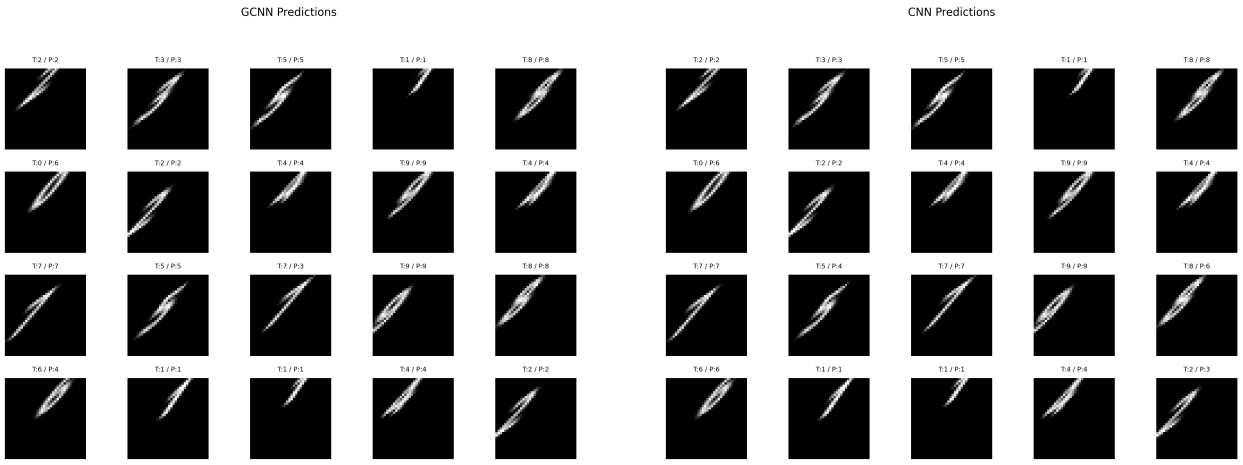


Figure 4: Accuracy comparison under affine transformation $A = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$. GCNN outperforms CNN with higher mean accuracy (0.80 vs. 0.72).



(a) Group Convolution Network (b) Standard CNN

Figure 5: Prediction comparison under affine transformation $A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$. G-CNN outperforms CNN with higher mean accuracy (0.6950 vs. 0.3150).



(a) Group Convolution Network (b) Standard CNN

Figure 6: Prediction comparison under affine transformation $A = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$. GCNN outperforms CNN with higher mean accuracy (0.80 vs. 0.72).

(a) Group Convolution Network (b) Standard CNN

Figure 7: Prediction comparison under affine transformation $A = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$. GCNN outperforms CNN with higher mean accuracy (0.8150 vs. 0.5950).

of transformations such as rotations, scalings, and translations, significantly broadening the class of transformations for which G-CNNs are theoretically justified. Our findings represent the first rigorous validation of G-CNN architectures in such a general affine setting. Furthermore, we showed that the complex group convolutions required for these networks can be simplified to standard integrals over $\mathbb{R}^2$, enhancing computational feasibility. These theoretical insights support the design of more robust and invariant learning models.

Beyond theoretical contributions, our simulations confirm that G-CNNs can outperform conventional CNNs, particularly in data-scarce scenarios. This highlights the practical benefits of incorporating affine invariance into deep learning architectures, potentially expanding their applicability to more complex and realistic pattern recognition tasks where geometric variability is prevalent.

References

Oludare Isaac Abiodun, Aman Jantan, Abiodun Esther Omolara, Kemi Victoria Dada, Nachaat AbdElatif Mohamed, and Humaira Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), 2018.

Parastoo Akbari, Matthew Gabriel, and Cameron A. MacKenzie. Retrieving and disseminating information about disasters through natural language processing tools. In *Proceedings of the IISE Annual Conference*, pp. 1–6. Institute of Industrial and Systems Engineers (IISE), 2022.

V. Andrearczyk, J. Fageot, V. Oreiller, X. Montet, and A. Depeursinge. Exploring local rotation invariance in 3d cnns with steerable filters. In *Int. Conf. on Med. Imaging with Deep Learning*, pp. 15–26. PMLR, 2019.

E. Bekkers. B-spline cnns on lie groups. *arXiv:1909.12057*, 2019.

E. Bekkers, R. Duits, and M. Loog. Training of templates for object recognition in invertible orientation scores: Application to optic nerve head detection in retinal images. In *Int. Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition*, pp. 464–477. Springer, 2015.

R. Bracewell, K. Chang, A. Jha, and Y. Wang. Affine theorem for two-dimensional fourier transform. *Electronics Lett.*, 29(3):304–304, 1993.

H. Chen, S. Liu, W. Chen, H. Li, and R. Hill. Equivariant point network for 3d point cloud analysis. In *Proc. IEEE CVF Conf. on Computer Vision and Pattern Recognition*, pp. 14514–14523, 2021.

T. Cohen and M. Welling. Group equivariant convolutional networks. In M. F. Balcan and K. Q. Weinberger (eds.), *Proc. 33rd Int. Conf. Mach. Learn.*, volume 48 of *Proc. Mach. Learn. Res.*, pp. 2990–2999, New York, NY, USA, 20–22 Jun 2016. PMLR.

T. Cohen, M. Geiger, J. Kohler, and M. Welling. Spherical cnns. *arXiv:1801.10130*, 2018.

S. Dieleman, J. De Fauw, and K. Kavukcuoglu. Exploiting cyclic symmetry in convolutional neural networks. In *Int. Conf. on Machine Learning*, pp. 1889–1898. PMLR, 2016.

Z. Eshkuvatov, N. Long, H. Midi, and A. Khaldjigitov. Polynomial spline approach for double integrals with algebraic singularity. In *J. Phys.: Conf. Ser.*, volume 435, pp. 012009. IOP Publishing, 2013.

R. Fisher, S. Perkins, A. Walker, and E. Wolfart. Affine transformation. `https://homepages.inf.ed.ac.uk/rbf/HIPR2/affine.htm`, 2000.

X. Guo, E. Zhu, X. Liu, and J. Yin. Affine equivariant autoencoder. In *IJCAI*, pp. 2413–2419, 2019.

I. Herstein. *Topics in Algebra*. John Wiley Sons, 1991.

E. Hoogeboom, J. W. T. Peters, T. S. Cohen, and M. Welling. Hexaconv. *arXiv:1803.02108*, 2018.

D. Knigge, D. Romero, and E. Bekkers. Exploiting redundancy: Separable group convolutional networks on lie groups. In *Int. Conf. on Machine Learning*, pp. 11359–11386. PMLR, 2022.

R. Kondor and S. Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Int. Conf. on Machine Learning*, pp. 2747–2755. PMLR, 2018.

R. Milad and K. F. Taylor. Harmonic analysis on the affine group of the plane. *J. Fourier Anal. Appl.*, 29 (3):30, 2023.

Ali Mohaddes and Johannes Lederer. Cardinality sparsity: Applications in matrix-matrix multiplications and machine learning. *Transactions on Machine Learning Research*, 2025.

Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

E. Oyallon and S. Mallat. Deep roto-translation scattering for object classification. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 2865–2873, 2015.

W. Schindler. Iwasawa's theorem and integrals on lie groups. *Math. Nachrichten*, 162(1):315–327, 1993.

Mohamed A. Shahin, Mark B. Jaksa, and Holger R. Maier. Artificial neural network applications in geotechnical engineering. *Australian Geomechanics*, 36(1):49–62, 2001.

B. Smets, J. Portegies, E. Bekkers, and R. Duits. Pde-based group equivariant convolutional neural networks. *J. Math. Imaging Vision*, 65(1):209–239, 2023.

N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds. *arXiv:1802.08219*, 2018.

Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

Y. Wang and M. Lu. An optimized system to solve text-based captcha. *arXiv:1806.07202*, 2018.

M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *Adv. Neural Inf. Process. Syst.*, 31, 2018.

Bernard Widrow, David E. Rumelhart, and Michael A. Lehr. Neural networks: Applications in industry, business and science. *Communications of the ACM*, 37(3):93–106, 1994.

M. Winkels and T. Cohen. 3d gcnns for pulmonary nodule detection. *arXiv:1804.04656*, 2018.

D. Worrall and G. Brostow. Cubenet: Equivariance to 3d rotation and translation. In *Proc. Eur. Conf. on Computer Vision (ECCV)*, pp. 567–584, 2018.

D. Worrall, S. Garbin, D. Turmukhambetov, and G. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 5028–5037, 2017.

Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Oriented response networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 519–528, 2017.

A Technical Results and Proofs

In this section, we initially present an example analogous to Example 5 but within the context of G_2. Then we provide the proofs of the main theorems in the section.

Example 6. *Consider the below input function:*

$$\Phi(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} \in [t_1, t_2] \times [s_1, s_2] \\ 0 & \text{otherwise} \end{cases},$$

then according to Example (4) for the lifting of Φ we have

$$(\mathcal{K}\Phi)[\boldsymbol{x}, \boldsymbol{A}] = \begin{cases} \frac{1}{|\det(\boldsymbol{A})|} & \text{if } \boldsymbol{x} \in [t_1, t_2] \times [s_1, s_2] \\ 0 & \text{otherwise} \end{cases},$$

as a result

$$\begin{aligned}
&\int_{G_2} \Phi([\boldsymbol{x}, \boldsymbol{A}]) k([\boldsymbol{y}, \boldsymbol{B}]^{-1}[\boldsymbol{x}, \boldsymbol{A}]) \mathrm{d}\mu_{G_2} \\
&= \int_{G_2} \Phi([\boldsymbol{x}, \boldsymbol{A}]) k\big(\boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{A}\boldsymbol{B}^{-1}\big) \mathrm{d}\mu_{G_2} \\
&= \int_{G_2} \frac{1}{|\det(\boldsymbol{A})|} k\big(\boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{A}\boldsymbol{B}^{-1}\big) \mathrm{d}\mu_{G_2} \\
&= \int_{t_1}^{t_2} \int_{s_1}^{s_2} \int_{\mathrm{GL}_2} \frac{1}{|\det(\boldsymbol{A})|^2} k(\boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{A}\boldsymbol{B}^{-1}) \mathrm{d}\mu_{\mathrm{GL}_2} \mathrm{d}x_1 \mathrm{d}x_2,
\end{aligned}$$

We define the separable kernel as follows

$$k(\boldsymbol{x}, \boldsymbol{P}) = |\det(\boldsymbol{P})|^2 \exp(\langle \boldsymbol{x}, \boldsymbol{l} \rangle) \mathcal{N}(\boldsymbol{P}),$$

where $\boldsymbol{l} = [l_1, l_2]$, and $\mathcal{N}(\boldsymbol{P})$ is the normalized heat kernel of $\mathrm{GL}_2(\mathbb{R})$. By this choice we obtain

$$\begin{aligned}
&\int_{t_1}^{t_2} \int_{s_1}^{s_2} \int_{\mathrm{GL}_2(\mathbb{R})} \frac{1}{|\det(\boldsymbol{A})|^2} k(\boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{A}\boldsymbol{B}^{-1}) \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}\mu_{\mathrm{GL}_2(\mathbb{R}} \\
&= \int_{t_1}^{t_2} \int_{s_1}^{s_2} \int_{\mathrm{GL}_2(\mathbb{R}} \frac{1}{|\det(\boldsymbol{A})|^2} |\det(\boldsymbol{B^{-1}})|^2 |\det(\boldsymbol{A})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l} \rangle) \mathcal{N}(\boldsymbol{P}) \mathrm{d}x_1 \mathrm{d}x_2 \mathrm{d}\mu_{\mathrm{GL}_2(\mathbb{R}} \\
&= \int_{t_1}^{t_2} \int_{s_1}^{s_2} \frac{1}{|\det(\boldsymbol{B})|^2} \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x} - \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l} \rangle) \int_{\mathrm{GL}_2(\mathbb{R}} \mathcal{N}(\boldsymbol{P}) \mathrm{d}\mu_{\mathrm{GL}_2(\mathbb{R}} \mathrm{d}x_1 \mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l} \rangle)} \int_{t_1}^{t_2} \int_{s_1}^{s_2} \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x}, \boldsymbol{l} \rangle) \mathrm{d}x_1 \mathrm{d}x_2
\end{aligned}.$$

Assume that $\boldsymbol{B}^{-1} = \begin{pmatrix} \beta_1 & \beta_2 \\ \beta_3 & \beta_4 \end{pmatrix}$, then we obtain

$$
\begin{aligned}
&\frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)} \int_{t_1}^{t_2}\int_{s_1}^{s_2} \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x}, \boldsymbol{l}\rangle)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)} \int_{t_1}^{t_2}\int_{s_1}^{s_2} \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x}, \boldsymbol{l}\rangle)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)} \int_{t_1}^{t_2}\int_{s_1}^{s_2} \exp(l_1\beta_1 x_1 + l_1\beta_2 x_2 + l_2\beta_3 x_1 + l_2\beta_4 x_2)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)} \int_{t_1}^{t_2}\int_{s_1}^{s_2} \exp\Big(x_1(l_1\beta_1 + l_2\beta_3) + x_2(l_1\beta_2 + l_2\beta_4)\Big)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)(l_1\beta_1 + l_2\beta_3)(l_1\beta_2 + l_2\beta_4)} \Big(\exp\Big(t_2(l_1\beta_1 + l_2\beta_3) - \exp\Big(t_1(l_1\beta_1 + l_2\beta_3)\Big) \\
&\times \Big(\exp\Big(s_2(l_1\beta_2 + l_2\beta_4)\Big) - \exp\Big(s_1(l_1\beta_2 + l_2\beta_4)\Big)\Big).
\end{aligned}
$$

For simplicity we can assume $l_1 = l_2 = 1$, then we have

$$
\begin{aligned}
&\frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, \boldsymbol{l}\rangle)} \int_{t_1}^{t_2}\int_{s_1}^{s_2} \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{x}, \boldsymbol{l}\rangle)\mathrm{d}x_1\mathrm{d}x_2 \\
&= \frac{1}{|\det(\boldsymbol{B})|^2 \exp(\langle \boldsymbol{B}^{-1}\boldsymbol{y}, [1,1]\rangle)(\beta_1 + \beta_3)(\beta_2 + \beta_4)} \Big(\exp\Big(t_2(\beta_1 + \beta_3) - \exp\Big(t_1(\beta_1 + \beta_3)\Big) \\
&\times \Big(\exp\Big(s_2(\beta_2 + \beta_4)\Big) - \exp\Big(s_1(\beta_2 + \beta_4)\Big)\Big).
\end{aligned}
$$

Proof of Theorem 2

Proof. We know that $\int_{\mathbb{R}^2} \frac{k(g^{-1}\boldsymbol{x})\Phi(\boldsymbol{x})}{|\det h|}\mathrm{d}\boldsymbol{x} = \int_{\mathbb{R}^2} k(\boldsymbol{x})\Phi(g\boldsymbol{x})\mathrm{d}\boldsymbol{x}$, then we have

$$
\begin{aligned}
&\sup_{g'}\Big|\big((\mathcal{K}\Phi_1) - \rho\left(g^{-1}\right)(\mathcal{K}\Phi_1)\big)\left(g'\right)\Big| \\
&= \sup_{g'}\left|\int_{\mathbb{R}^2} k(\boldsymbol{x})\Phi_1(g'\boldsymbol{x})d\boldsymbol{x} - k(\boldsymbol{x})\Phi_1(gg'\boldsymbol{x})\mathrm{d}\boldsymbol{x}\right| \\
&\leq \sup_{g'}\int_{\mathbb{R}^2} |k(\boldsymbol{x})||\Phi_1(g'\boldsymbol{x}) - \Phi_1(gg'\boldsymbol{x})|\mathrm{d}\boldsymbol{x},
\end{aligned}
$$

by setting $g'\boldsymbol{x} = \boldsymbol{y}$ for the last term in above we have

$$
\sup_{g'}\int_{\mathbb{R}^2} |k(\boldsymbol{x})||\Phi_1(g'\boldsymbol{x}) - \Phi_1(gg'\boldsymbol{x})|\mathrm{d}\boldsymbol{x} \leq \epsilon\int_{\mathbb{R}^2} |k(\boldsymbol{x})|\mathrm{d}\boldsymbol{x} = \epsilon\|k\|_1^{\mathbb{R}^2}.
$$

□

Proof of Theorem 3

Proof. We have

$$
\begin{aligned}
\|(\mathcal{K}\Phi_1) * k - \rho(\tilde{h})(\mathcal{K}\Phi_2) * k\|_{\sup}^{G_2} &= \sup\Big|\int_{G_2} (\mathcal{K}\Phi_1)(g)k(h^{-1}(g)) - \rho(\tilde{h})(\mathcal{K}\Phi_2)(g)k(h^{-1}g)\mathrm{d}\mu_{G_2}(g)\Big| \\
&\leq \sup\int_{G_2}\Big|(\mathcal{K}\Phi_1)(g)k(h^{-1}(g)) - \rho(\tilde{h})(\mathcal{K}\Phi_2)(g)k(h^{-1}g)\Big|\mathrm{d}\mu_{G_2}(g) \\
&\leq \sup\int_{G_2}\Big|(\mathcal{K}\Phi_1)(g) - \rho(\tilde{h})(\mathcal{K}\Phi_2)(g)\Big|\Big|k(h^{-1}(g))\Big|\mathrm{d}\mu_{G_2}(g) \\
&\leq \epsilon\|k\|_1^{G_2}.
\end{aligned}
$$

The second part of the theorem results by selecting $k = \delta(g - h')$. □

Proof of Theorem 4

Proof. We know that

$$\Big| \int_{G_2} \big((\mathcal{K}\Phi_1) * k - (\mathcal{K}\Phi_2) * k\big)(h) \mathrm{d}\mu_{G_2}(h) \Big| =$$
$$\Big| \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_1)(g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) - \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \Big|.$$

Then for the second term in the above equation we have and replacing g with $\tilde{h}^{-1}g$ we have

$$\begin{aligned}
&\int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(g) k(h^{-1}g)\big) d\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \\
&= \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k(h^{-1}\tilde{h}^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \\
&= \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k((\tilde{h}h)^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h),
\end{aligned}$$

if we set

$$\Phi(h) = \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k((\tilde{h}h)^{-1}g)\big) \mathrm{d}\mu_{G_2}(g),$$

then

$$\begin{aligned}
&\int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k((\tilde{h}h)^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \\
&= \int_{G_2} \Phi(h) \mathrm{d}\mu_{G_2}(h) = \int_{G_2} \Phi(\tilde{h}h) \mathrm{d}\mu_{G_2}(h) \\
&= \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h),
\end{aligned}$$

therefore,

$$\begin{aligned}
&\Big| \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_1)(g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) - \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \Big| \\
&= \Big| \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_1)(g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) - \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_2)(\tilde{h}^{-1}g) k(h^{-1}g)\big) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \Big| \\
&= \Big| \int_{G_2} \int_{G_2} \big((\mathcal{K}\Phi_1)(g) - (\mathcal{K}\Phi_2)(\tilde{h}^{-1}g)\big) k(h^{-1}g) \mathrm{d}\mu_{G_2}(g) \mathrm{d}\mu_{G_2}(h) \Big| \\
&= \Big| \int_{G_2} \big((\mathcal{K}\Phi_1) - (\mathcal{K}\Phi_2) \circ \tilde{h}^{-1}\big) * k \, \mathrm{d}\mu_{G_2}(h) \Big| \\
&\leq \int_{G_2} \Big| \big((\mathcal{K}\Phi_1) - (\mathcal{K}\Phi_2) \circ \tilde{h}^{-1}\big) * k \Big| \mathrm{d}\mu_{G_2}(h) \\
&= \Big\| \big((\mathcal{K}\Phi_1) - (\mathcal{K}\Phi_2) \circ \tilde{h}^{-1}\big) * k \Big\|_1^{G_2} \leq \epsilon \|k\|_1^{G_2}.
\end{aligned}$$

□

Proof of Theorem 5

Proof. We know that

$$\begin{aligned}&\int_{G_2}\Phi([\boldsymbol{x},\boldsymbol{A}])k([\boldsymbol{y},\boldsymbol{B}]^{-1}[\boldsymbol{x},\boldsymbol{A}])\mathrm{d}\mu_{G_2}\\&=\int_{G_2}\Phi([\boldsymbol{x},\boldsymbol{A}])k\big(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y},\boldsymbol{A}\boldsymbol{B}^{-1}\big)\mathrm{d}\mu_{G_2}.\end{aligned}$$

Employing (6) we have

$$\begin{aligned}&\int_{G_2}\Phi([\boldsymbol{x},\boldsymbol{A}])k(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y},\boldsymbol{A}\boldsymbol{B}^{-1})\mathrm{d}\mu_{G_2}\\&=\int_{\mathrm{GL}_2(\mathbb{R})}\int_{\mathbb{R}^2}\Phi[\boldsymbol{x},\boldsymbol{A}]k(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y},\boldsymbol{A}\boldsymbol{B}^{-1})\frac{\mathrm{d}x_1\mathrm{d}x_2}{|\det(\boldsymbol{A})|}\mathrm{d}\mu_{\mathrm{GL}_2},\end{aligned}$$

we also set

$$H_{\Phi,k}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{y}):=\int_{\mathbb{R}^2}\Phi[\boldsymbol{x},\boldsymbol{A}]k(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y},\boldsymbol{A}\boldsymbol{B}^{-1})\frac{\mathrm{d}x_1\mathrm{d}x_2}{|\det(\boldsymbol{A})|}. \tag{9}$$

From separability property of kernel we have $k(\boldsymbol{x},\boldsymbol{A})=k_1(\boldsymbol{x})k_2(\boldsymbol{A})$. As a result

$$\begin{aligned}H_{\Phi,k}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{y})&=\int_{\mathbb{R}^2}\Phi[\boldsymbol{x},\boldsymbol{A}]k_1(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y})k_2(\boldsymbol{A}\boldsymbol{B}^{-1})\frac{\mathrm{d}x_1\mathrm{d}x_2}{|\det(\boldsymbol{A})|}\\&=\frac{k_2(\boldsymbol{A}\boldsymbol{B}^{-1})}{|\det(\boldsymbol{A})|}\int_{\mathbb{R}^2}\Phi[\boldsymbol{x},A]k_1(\boldsymbol{B}^{-1}\boldsymbol{x}-\boldsymbol{B}^{-1}\boldsymbol{y})\mathrm{d}x_1\mathrm{d}x_2\\&=\frac{k_2(\boldsymbol{A}\boldsymbol{B}^{-1})}{|\det(\boldsymbol{A})|}\Big(\Phi*(k_1\circ\boldsymbol{B}^{-1})\Big)\\&=\frac{k_2(\boldsymbol{A}\boldsymbol{B}^{-1})}{|\det(\boldsymbol{A})|}\mathcal{F}^{-1}\Big(\mathcal{F}(\Phi)\mathcal{F}(k_1\circ\boldsymbol{B}^{-1})\Big),\end{aligned} \tag{10}$$

where $\mathcal{F}(\cdot)$ denotes the Fourier transform. The next step is to find an explicit form for the Fourier transform. We can apply the result from (Bracewell et al., 1993). Assume that $\mathcal{F}(K_1)=\widehat{K}_1(\boldsymbol{u})$ and $\mathcal{F}(\Phi)=\widehat{\Phi}(\boldsymbol{u})$ then we have

$$H_{\Phi,K}(a,b,c,d,\boldsymbol{B},\boldsymbol{y})=\frac{K_2(\boldsymbol{A}\boldsymbol{B}^{-1})}{|\det(\boldsymbol{A})||\det(\boldsymbol{B}^{-1})|}\mathcal{F}^{-1}\Big(\widehat{\Phi}(\boldsymbol{u})\widehat{K}_1(\boldsymbol{B}^{\top}\boldsymbol{u})\Big).$$

Now we use decomposition of $\mathrm{GL}_2(\mathbb{R})$ as $K_0\ltimes H(1,0)$ in (Milad & Taylor, 2023; Schindler, 1993).

Proposition 1 (Proposition 5.1 of (Milad & Taylor, 2023))**.** *If* $\boldsymbol{A}=\begin{pmatrix}a&b\\c&d\end{pmatrix}\in\mathrm{GL}_2(\mathbb{R})$*, then* $\boldsymbol{A}$ *can be uniquely decomposed as the product* $\boldsymbol{A}=\boldsymbol{M}_{\boldsymbol{A}}\boldsymbol{C}_{\boldsymbol{A}}$ *with* $\boldsymbol{M}_{\boldsymbol{A}}\in K_0$ *and* $\boldsymbol{C}_{\boldsymbol{A}}\in H_{(1,0)}$*. In fact*

$$\boldsymbol{M}_{\boldsymbol{A}}=\begin{pmatrix}s&-t\\t&s\end{pmatrix},\ \textit{with}\ \ s=\frac{d(ad-bc)}{b^2+d^2},t=\frac{-b(ad-bc)}{b^2+d^2},$$

and

$$\boldsymbol{C}_{\boldsymbol{A}}=\begin{pmatrix}1&0\\u&v\end{pmatrix},\ \textit{with}\ \ u=\frac{cd+ab}{(ad-bc)},v=\frac{b^2+d^2}{(ad-bc)}.$$

This factorization leads to a parallel factorization of G_2*.*

Consider the one to one transform between H and H^* so that $H^*(s,t,u,v,\boldsymbol{B},\boldsymbol{y}) := H_{f,k}(a,b,c,d,\boldsymbol{B},\boldsymbol{y})$, where $a = s - ut$, $c = t + us$, $b = -t/v$, and $d = s/v$. Employing the above proposition and Equation (7) we can write

$$\begin{aligned} H'(\boldsymbol{B},\boldsymbol{y}) &= \int_{\mathrm{GL}_2(\mathbb{R})} H_{\Phi,k}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{y})\mathrm{d}\mu_{\mathrm{GL}_2} \\ &= \int_{\mathrm{GL}_2(\mathbb{R})} H^*(s(a,b,c,d),t(a,b,c,d),u(a,b,c,d),v(a,b,c,d),\boldsymbol{B},\boldsymbol{y})\mathrm{d}\mu_{\mathrm{GL}_2}. \end{aligned}$$

Therefore, we obtain

$$\int_{\mathrm{GL}_2(\mathbb{R})} H_{\Phi,k}(\boldsymbol{A},\boldsymbol{B},\boldsymbol{y})\mathrm{d}\mu_{\mathrm{GL}_2} = \int_{K_0}\int_{H_{(1,0)}} H^*(s,t,u,v,\boldsymbol{B},\boldsymbol{y})|v|\mathrm{d}\mu_{H_{(1,0)}}\mathrm{d}\mu_{K_0},$$

as $\det(\boldsymbol{C_A}) = |v|$. Then we define

$$\begin{aligned} H^*(s,t,\boldsymbol{B},\boldsymbol{y}) &= \int_{H_{(1,0)}} H^*(s,t,u,v,\boldsymbol{B},\boldsymbol{y})\det(\boldsymbol{C_A})\mathrm{d}\mu_{H_{(1,0)}}(u,v) \\ &= \int_{G_1} H^*(s,t,u,v,\boldsymbol{B},\boldsymbol{y})\det(\boldsymbol{C_A})\mathrm{d}\mu_{G_1}(u,v) \\ &= \int_{\mathbb{R}}\int_{\mathbb{R}} H^*(s,t,u,v,\boldsymbol{B},\boldsymbol{y})\frac{\mathrm{d}u\mathrm{d}v}{|v|}. \end{aligned}$$

The next step is to compute integration of $H^*(s,t,\boldsymbol{B},\boldsymbol{y})$ over K_0, which is equal to

$$\int_{\mathbb{R}}\int_{\mathbb{R}} H^*(s,t,\boldsymbol{B},\boldsymbol{y})\frac{\mathrm{d}s\mathrm{d}t}{s^2+t^2}. \tag{11}$$

□