Exploring RIC-CNN for rotation dependent images

CHAN, Chun Yu
cychandp@connect.ust.hk

FUNG, Hei Yuen
hyfungaj@connect.ust.hk

HE, Ka Sing
ksheaa@connect.ust.hk

Abstract

Convolutional Neural Networks (CNNs) have demonstrated remarkable performance in various computer vision tasks, attributed to their ability to learn hierarchical features and their translation equivariance. However, standard CNNs lack rotational invariance, leading to performance degradation with rotated objects. While Rotation-Invariant Convolutional Neural Networks (RIC-CNNs) address this, they exhibit limitations in discriminating between rotationally symmetric objects with distinct meanings at different orientations (e.g., '6' and '9'). This limitation, as noted in Mo et al. (2022), can hinder performance in scenarios where such distinctions are crucial. In this work, we explore combining the rotational invariance of RIC-CNNs with the discriminative power of standard CNNs. Specifically, we investigate several approaches to integrate these architectures, aiming to achieve robust performance on both rotated and non-rotated image datasets. We evaluate our approaches on the MNIST and Traffic Signs datasets, demonstrating improved performance and robustness compared to using either CNNs or RIC-CNNs alone.

1. Introduction

Convolutional Neural Networks (CNNs) have achieved state-of-the-art results in numerous computer vision tasks, owing to their ability to learn hierarchical features and their translation equivariance. However, standard CNNs are inherently sensitive to image rotations, leading to degraded performance when processing rotated images.

1.1. Deficiencies of CNNs and the Role of Rotational Invariance

The lack of rotational invariance in CNNs necessitates the use of data augmentation techniques, where training datasets are expanded with rotated versions of existing images. While this can improve performance, it introduces computational overhead and requires careful tuning to determine the appropriate number of rotated versions. To address this, Rotation-Invariant Convolutional Neural Networks (RIC-CNNs) [**?**] have been proposed, utilizing rotation-invariant coordinate systems to achieve invariance to arbitrary rotations.

1.2. Limitations of RIC-CNNs and the Need for Equivariance

While RIC-CNNs effectively handle rotated images, their strict rotational invariance can be a limitation. In scenarios where distinguishing between rotated versions of objects is crucial (e.g., '6' vs. '9'), RIC-CNNs may struggle. In such cases, rotational equivariance – where feature maps transform in the same way as the input image – is desirable to preserve rotational information.

1.3. Our Hybrid Approach

To balance the strengths and weaknesses of CNNs and RIC-CNNs, we propose a hybrid approach that combines both architectures. This allows us to leverage the discriminative power of CNNs for rotation-sensitive tasks while benefiting from the robustness of RIC-CNNs to handle general image rotations. We explore several integration strategies to effectively fuse features from both networks.

1.4. Contributions

Our contributions include:

- Developing hybrid CNN-RIC-CNN architectures for improved performance on both rotated and non-rotated images.
- Evaluating the proposed methods on the MNIST and Traffic Signs datasets.

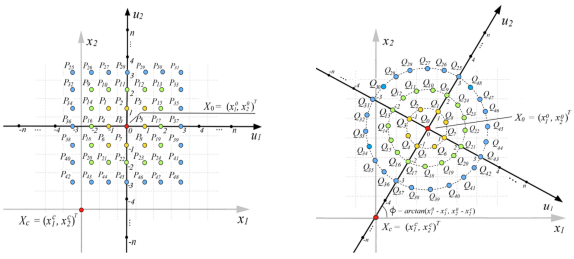
2. Related Works

In this section, we focus on Rotation-Invariant Convolutional Neural Networks (RIC-CNNs).

2.1. Rotation-Invariant Coordinate Convolutional Neural Network

Mo and Zhao (2022) introduce the *Rotation-Invariant Coordinate Convolution* (RIC-C) to embed structural rotation invariance into the convolution operation. We first recall the standard convolution:

$$\Phi_C(X_0, F) \;=\; \sum_{P \in S} W(P)\, F(X_0 + P), \qquad (1)$$



(a) The order of $(4n^2+4n+1)$ sample points P on the regular grid $\mathcal{S}$.

(b) The order of $(4n^2+4n+1)$ sample points Q on the rotation-invariant coordinate system $\mathcal{T}$.

Figure 1. Two sets of sampling points used for calculating Φ_C and $\Phi_{\mathrm{RIC}-C}$, respectively. It is clear that $Q_\alpha \in \mathcal{T}$ can be regarded as the corresponding point of $P_\alpha \in \mathcal{S}$, where $\alpha = 0, 1, 2, \ldots, 4n^2 + 4n$.

where S is the usual fixed sampling grid (e.g. a 3×3 neighborhood) and $W(P)$ are the kernel weights.

RIC-C replaces the fixed grid S by a *rotation-aware* sampling pattern $\mathcal{T}_{X_0}$. Define

$$\varphi \;=\; atan2\big(x_2^0,\, x_1^0\big) \quad \text{for} \quad X_0 = (x_1^0, x_2^0)^T$$

and let the kernel cover radii $r = 1, \ldots, n$. On each circle of radius r, sample

$$Q_{r,i} = \big(r\cos(\varphi + \tfrac{2\pi\, i}{8r}),\; r\sin(\varphi + \tfrac{2\pi\, i}{8r})\big), \quad i = 0, 1, \ldots, 8r-1. \tag{2}$$

Collecting all such $Q_{r,i}$ (plus the center $Q_0 = (0,0)$) defines $\mathcal{T}_{X_0} = \{Q_{r,i}\}$. The RIC-C operation is then

$$\Phi_{\text{RIC-C}}(X_0, F) \;=\; \sum_{Q \,\in\, \mathcal{T}_{X_0}} W(Q)\, F(X_0 + Q). \tag{3}$$

Because $\mathcal{T}_{X_0}$ itself rotates by the same angle when the input is rotated about the image center, one can show $\Phi_{\text{RIC-C}}(Y_0, G) = \Phi_{\text{RIC-C}}(X_0, F)$ for any rotation $G(Y) = F(R_{-\theta}Y)$, proving invariance. In **??**, the sampling scheme of each method can be compared.

Key Limitations

- **Loss of Orientation Information.** Enforcing full invariance discards absolute angle cues, making it hard to distinguish classes that differ only by rotation (e.g. '6' vs. '9').
- **Center-Only Invariance.** The proof assumes rotations about the image (or feature-map) center; arbitrary pivot points are not covered.
- **Interpolation & Pooling Artifacts.** Sampling at fractional coordinates requires interpolation (e.g. bilinear), and subsequent max-pooling layers are not strictly equivariant, leading to small but accumulative errors.

3. Methodology

In this section, we describe the three methods we employ to combine CNNs and RIC-CNNs:

3.1. Double Branch Model

In this approach, we create a network with two branches: one with a standard CNN architecture, and the other with a RIC-CNN architecture. The idea is to allow each branch to learn different aspects of the input image. The CNN branch learns the detailed features, and the RIC-CNN branch learns the rotationally invariant features. The outputs of the two branches are then combined to make the final prediction. In our implementation, the RIC-CNN branch is frozen for the first few epochs of training. This allows the CNN branch to initially learn basic features without being influenced by the RIC-CNN branch.

3.2. Take Best Confidence

This approach is based on the intuition that when a rotated image is passed through a standard CNN, the CNN may not recognize the pattern if it has not seen that specific rotation during training. This leads to a more spread-out confidence score across different classes, resulting in a lower peak confidence. RIC-CNN, on the other hand, is expected to produce a more reliable prediction with a higher peak confidence due to its rotational invariance.

Similarly, when an unrotated image is passed through an RIC-CNN, examples like label 6 and 9 in MNIST will look the same for RIC-CNN because of its rotational invariance property. This leads to a more spread-out confidence score across these classes, resulting in a lower peak confidence. In this case, standard CNN can clearly distinguish the difference between these labels by relying on the rotational information and give more concentrated and higher peak confidence compared to RIC-CNN.

Therefore, in this approach, we take the prediction with the highest confidence score from either the CNN or the RIC-CNN to be the output.

3.2.1. Notation

Let $I \in \mathbb{R}^{H\times W\times C}$ be the input image, $\mathbf{y}_c \in \mathbb{R}^{C_l} = f_c(I, \hat{\theta}_c)$ and $\mathbf{y}_r \in \mathbb{R}^{C_l} = f_r(I, \hat{\theta}_r)$, where $\mathbf{y}_c$ and $\mathbf{y}_r$ are the prediction logits of the standard CNN f_c and RIC-CNN f_r respectively, where C_l is the number of labels of the dataset of the classification task, with their corresponding weights $\hat{\theta}_c$ and $\hat{\theta}_r$.

3.2.2. Independent Branch Training

Both standard CNN branch and RIC-CNN branch are trained independently using the same datasets at the same

time with the combined loss function

$$\mathcal{L} = \mathcal{L}_{CE}(\sigma_s(\mathbf{y}_c), \mathbf{y}) + \mathcal{L}_{CE}(\sigma_s(\mathbf{y}_r), \mathbf{y})$$

where $\mathcal{L}_{CE}$ denote the cross-entropy loss function, σ_s denote the softmax function, and $\mathbf{y} \in \mathbb{R}^{C_l}$ is the one-hot representation of the ground truth label.

The *Take Best Confidence* will not be used in the training stage to prevent training collapse to one model as only one branch of the model will the gradient of the loss flows through after taking maximum on confidence scores between the two.

3.2.3. Take Best Confidence Prediction

In prediction stage, both branches will contribute to the prediction to produce confidence scores $\mathbf{c}_c \in \mathbb{R}^{C_l}$ and $\mathbf{c}_r \in \mathbb{R}^{C_l}$ to each label class. We look for the maximum confidence score of each labels between the two branches, and select the label with the highest confidence as the final prediction output. More specifically, we determine the prediction output p by:

$$p = \arg\max_c(\mathbf{z}_c),\ \mathbf{z_i} = \max(\mathbf{c}_{c,i}, \mathbf{c}_{r,i}),\ i = 1...C_l$$

$$\mathbf{c}_j = \sigma_s(\mathbf{y}_j),\ j \in \{c, r\}$$

By comparing the elementwise maximum of the confidence scores by the two branches and taking the argmax of the two, we greedily select the desired class that has the global maximum confidence given by the two branches. This helps to find the most probable class label in both the said situations by believing in the branch that has more confidence on the result, either an unrotated image or a rotated image has been given.

3.3. Feature Fusion

RIC–CNN's intermediate feature maps are rotation-equivariant: when the input is rotated, the RIC branch's feature map F_r rotates correspondingly before global averaging. In contrast, a standard CNN's feature map F_c remains rotationally variant. This mismatch between F_r and F_c encodes rich rotational information. To harness this, we propose a *Feature Fusion* module that compares and dynamically weights these two feature maps to extract and leverage rotational cues for improved robustness.

3.3.1. Notation

Let $F_r \in \mathbb{R}^{H\times W\times C}$ be the feature map produced by the RIC branch and $F_c \in \mathbb{R}^{H\times W\times C}$ the feature map from the standard CNN branch at the same spatial resolution and channel dimensionality.

3.3.2. Weight Generation

We concatenate the two feature maps along the channel dimension and pass them through a 1×1 convolutional layer $g(\cdot)$, followed by a sigmoid activation to produce a spatially-varying weight map:

$$W = \sigma\big(g([F_r \| F_c])\big),$$

where $[\cdot\|\cdot]$ denotes channel-wise concatenation and σ is the element-wise sigmoid.

3.3.3. Fused Representation

The fused feature map F_f is computed as a per-pixel interpolation of F_r and F_c:

$$F_f = W \odot F_r + (1 - W) \odot F_c,$$

where $\odot$ denotes broadcasted element-wise multiplication. In regions where the RIC branch's rotation-equivariant features are more informative, W approaches 1; elsewhere, the standard CNN features dominate.

3.3.4. Auxiliary Supervision

To encourage each branch to learn discriminative features, we attach auxiliary classifiers $h_r(\cdot)$ and $h_c(\cdot)$ to F_r and F_c, respectively. The final classification is performed by $h_f(F_f)$. The overall loss is:

$$\mathcal{L} = \mathcal{L}_{\mathrm{CE}}\big(h_f(F_f), y\big) + \lambda_r\, \mathcal{L}_{\mathrm{CE}}\big(h_r(F_r), y\big) + \lambda_c\, \mathcal{L}_{\mathrm{CE}}\big(h_c(F_c), y\big),$$

where $\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss, y the ground-truth label, and λ_r, λ_c balance the auxiliary terms.

This fusion mechanism dynamically extracts rotational information from the mismatch between equivariant and variant feature maps, leading to balanced performance on both rotated and non-rotated inputs.

4. Results

In this section, we present the results of our experiments on the MNIST and Traffic Signs datasets.

4.1. MNIST Dataset

Our experiments on the MNIST dataset demonstrate the effectiveness of our proposed methods in handling rotated digits. The Double Branch model, Take Best Confidence, and Feature Fusion approaches all show improved performance compared to a standard CNN. In particular, the Feature Fusion method achieves the best results, demonstrating the benefit of combining both rotation-invariant and rotation-equivariant features. The results are shown in **??**.

4.2. Traffic Signs Dataset

The Traffic Signs dataset presents a more complex challenge due to the presence of rotational symmetries in some

Figure 2. Results on the MNIST dataset.

signs. Our methods also show promising results on this dataset. Notably, the Feature Fusion method again outperforms the other approaches, highlighting its ability to effectively handle images with more complex rotational properties. The results on the Traffic Signs dataset also confirm that our methods improve the robustness of CNNs to rotation in more complex, real-world scenarios.