Quasi-Conformal Transformer Network

Han Zhang*, Qiguang Chen*, Yuchen Guo*, and Lok Ming Lui*

Abstract. Information is not generally distributed uniformly in an image domain. Thus, to make the convolutional neural network focus more on those important, some deformation on convolution windows or feature maps should be applied. Besides, the topology of an image should be preserved from the ideas for defining the convolution operation. However, controlling topology is hard and not convenient for existing methods since they all use vector representation for displacements. In this paper, we proposed Quasi-Conformal Transformer Network using Beltrami representation, which is a strong representation to control the bijectivity and the degree of geometric deformations. Together with our Beltrami Solver Net(BSN), we proposed an end-to-end learnable network, which advantages other works on its power to control the geometric deformation of the feature maps.

Key words. deformable convolution, deformable pooling, disturblance-invariant, bijectivity

1. Introduction.

The information is not usually distributed uniformly in images. The target objects in images may differ in pose, position, and scale. Some human-caused errors may also bring challenges to image processing. To solve this problem, some feature representation methods that are invariant to transformation like SIFT (scale-invariant feature transform) [24] are proposed. With the development of computing power, the convolutional neural network (CNN) [17], with its transformation-invariant convolution windows, becomes the most popular and practical model that altered the landscapes of the computer vision community. However, since the convolution window and the pooling window are both defined to be in the fixed size and shape, some works accounting for adaptive convolution are proposed.

The adaptive convolution is mainly defined through two approaches. The spatial transformer (STN) [11] proposed the learnable spatial transformation on feature maps to focus only on regions that contain the most information. However, STN is not convenient to assign non-rigid transformations in the network. Though it may become possible with the thin-plate spline(TSP) method, the topology of the transformation is not guaranteed to be preserved. Another way is direct to define the deformable convolution(DeformConv) [6], where the displacement vectors are assigned directly on convolution windows. Since this work use vector representation, they are easy to result in a messy deformation like that in Figure.1 which contains self-intersections and failed to keep the topology of an image.

However, consideration for topology is a very important advantage of a convolutional neural network over a fully-connected neural network, which ignores it[17]. Reviewing the traditional sliding window methods, like the idea of Sobel operator [13], Prewitt operator [26], *et al.* , is to compute the gradient of the intensity function of images. Since the gradient computation will only involve neighbors, the operator window should always encircle a continuous region. More than that, if the displacement vectors are not restricted properly, overfitting in the process will easily occur. From these perspectives, the topology of images and feature maps should be preserved. In some cases, even the degree of the freedom of deformation for

*Department of Mathematics, The Chinese University of Hong Kong, Shatin, Hong Kong (hanzhang001@cuhk.edu.hk,ycguo@math.cuhk.edu.hk, im.ki@link.cuhk.edu.hk, lmlui@math.cuhk.edu.hk).

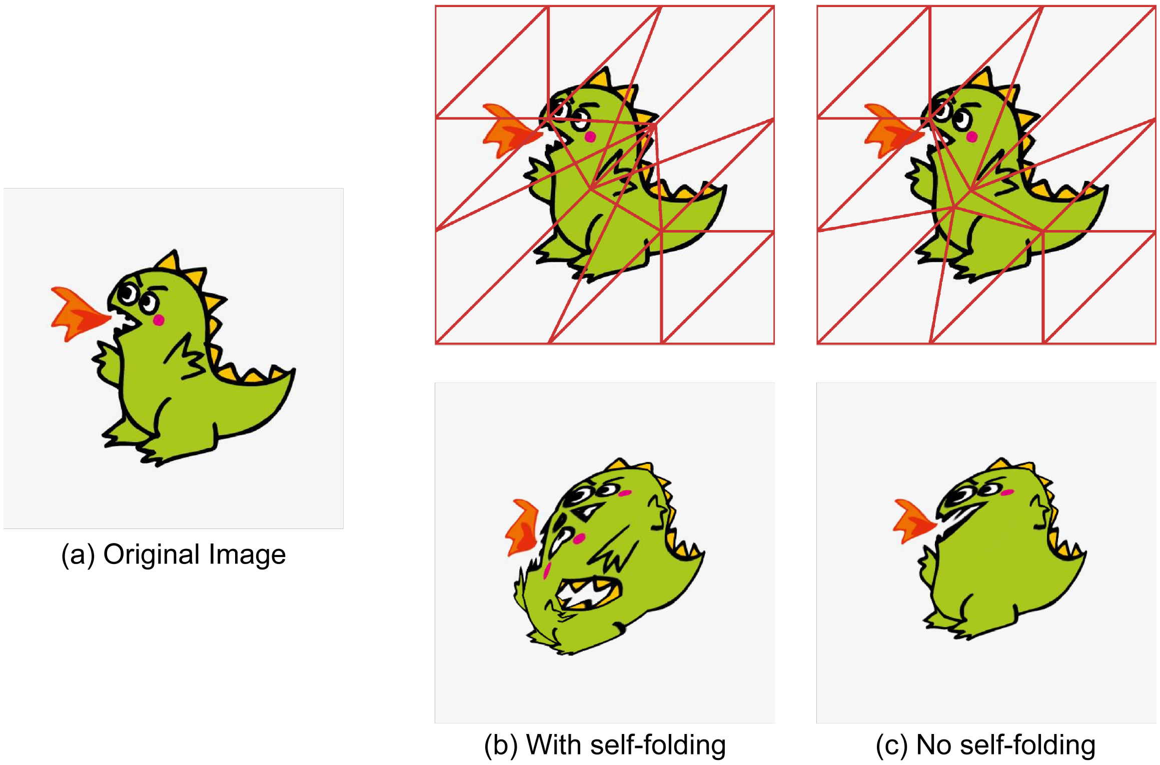

Figure 1: Example for images who failed to preserve the topology during processing. The dinosaur are warped to gain an extra mouth by fold its tail inside.

either feature map or convolution window should be controlled to reduce overfitting.

However, bijectivity is very important for tasks that want to preserve the topology and avoid self-intersection. If the transformation mapping failed to preserve the original topology, the transformed image would easily lose its original semantic meaning. The idea can be clearly explained by Figure.1, the dinosaur got another mouth in its body which is wrongly mapped by its tail. Thus, to assign reasonable spatial attention to the input images, we propose our quasi-conformal transformer network in this paper. Compared to the previous work[11], our model can produce pixel-wise transformation, which outperforms the TPS variant of spatial transformer network that failed to produce topology-preserving mapping.

Our model is based on the quasi-conformal theory and uses Beltrami coefficient as the mapping representation instead of control points or vector field. By restricting the Beltrami coefficient to be less than 1, the associated mapping can be guaranteed to be bijective. Thus, our quasi-conformal transformer network is composited of two modules (Figure.7). The coefficient estimator predicts the Beltrami coefficients that represent the mapping and the Beltrami solver network that transfers the Beltrami representation into vector representation that is convenient to do spatial transformation pixel-wisely. Individual modules will be introduced and discussed in more detail in Section.4.

To evaluate the capacity of the quasi-conformal transformer network, we test the classification results on images with different kinds of deformation. On such deformed images, conventional standard rectangle convolution can not accomplish the tasks well. However, due to the spatial attention assigned by the quasi-conformal transformer, our model can obtain much better results than a network without spatial transformers. Compared to the thin plate spline variant of spatial transformer network, our method can also acquire a higher accuracy rate and are easier to optimize because of the proper topology constraint.

To sum up, our contributions are:

- We introduced Beltrami representation for a pixel-wise spatial transformer in neural network, which is convenient to control the topologic property of the transformation mapping.
- By controling Beltrami coefficients, the mappings produced are guaranteed to be topology-preserving. By experiments results, such mapping help with classification results.
- We can achieve deformable convolution and deformable pooling with quasi-conformal transformer network, which is difficult by using the pixel-wise thin plate spline variant of spatial transformer network as it usually failed to preserve the topology of the transformation mapping.

2. Related Work.

2.1. Computational Quasi-Conformal. Computational conformal is a powerful tool to control the geometric variation and topology in image science[16, 22] and surface processing[20, 8]. Benefitting from the Beltrami representation, the mapping between two different domains can preserve good geometric properties like bijectivity and smoothness. Through controlling the Beltrami coefficients with such representation of mappings. Driven by the motivation to preserve different geometric information, ways of parameterization methods are proposed [9, 3, 2]. Such convenient representation are also popular and succeed in computational fabrication community [29, 5, 25]. With the capability to handle large deformations, the quasi-conformal method also succeed in registration for images [16, 22] and surfaces [4] and segmentation with topology- and convexity prior [33, 28].

2.2. Deep Learning. Neural network models [18] are widely employed and greatly successed in different fields like image science and natural language processing. With convolution neural network[17], translation invariants become available for learning methods and introduce the development of image science to the next stage. Classification is the first vision task that benefits from deep layers[15]. Inspired by the success, other tasks adopt deep learning models to solve the problems[]. Besides, the great success of the application of deep learning, the model itself is also developing. Max-pooling layers are introduced into the neural network model for dimensionality reduction by [30]. [27] increase the depth of the network by using very small (3×3) convolution windows to enhance the non-linearity of the network. With shortcut connections, ResNet [10] overcome the vanishing problem that prevents models from learning.

Jaderberg *et al.* [11] proposed the spatial transformer network, which is the first work to introduce spatial transformation to assign attention to feature map. Warping a feature map and doing convolution in the normal way is mathematically equivalent to defining some special convolutions learned from data. Jeon and Kim [12] proposed active convolution that can sample the locations of the convolution with some displacements that are shared over different spatial locations. However, the information is not uniformly distributed in the feature map. Not all pixels contribute equally to the final results[23]. Motivated by this, Qi *et al.* [6, 34] designed deformable convolution whose convolution differs not only among different maps but even on different spatial locations of the feature map.

3. Mathmetical Background.

In this chapter, we introduce some fundamental geometry concepts and theories that are related to our model. In brief words, a quasi-conformal mapping is mainly used for our registration map. Besides, since we need to compress our registration map as discussed in the introduction, Fourier compression for Beltrami representation will also be introduced.

3.1. Conformal Maps.

Definition 3.1. *(Quasi-conformal mape). A conformal map is a map $f : \mathbb{C} \to \mathbb{C}$ that satisfying the Beltrami equation*

$$\frac{\partial f}{\partial \bar{z}} = \mu(z) \frac{\partial f}{\partial z} \tag{3.1}$$

for some complex-valued function named as Beltrami coefficient μ satisfying $\|\mu\|_\infty < 1$ and $\frac{\partial f}{\partial z}$ is non-vanishing almost everywhere. The complex partial derivatives are given by

$$\frac{\partial f}{\partial z} := \frac{1}{2}\left(\frac{\partial f}{\partial x} - i\frac{\partial f}{\partial y}\right) \quad \text{and} \quad \frac{\partial f}{\partial \bar{z}} := \frac{1}{2}\left(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}\right) \tag{3.2}$$

μ is the Beltrami representation, which is also called the Beltrami coefficient, of the quasi-conformal map f. It's worthy to mention that μ is a measure of non-conformality. Particularly, for a point p, suppose $\mu(p) = 0$. Then, the associated quasi-conformal map f is conformal around a small neighborhood of p. In this case, Equation 3.1 is the Cauchy-Riemann equation. This can also illustrate that conformality analysis of a quasi-conformal map f can be simplified into the analysis of its associated Beltrami coefficient μ. Infinitesimally, such a map f can be rewritten as follows in a local neighborhood around a point p:

$$\begin{aligned} f(z) &= f(p) + f_z(p)z + f_{\bar{z}}(p)\bar{z} \\ &= f(p) + f_z(p)(z + \mu(p)\bar{z}) \end{aligned} \tag{3.3}$$

This further enhanced our discussion before that f is conformal when $\mu(p) = 0$. To explain for the equation above, $f(p)$ is a translation, while $f_z(p)$ is a dilation. Since both of them are conformal, all the non-conformality of f is brought by $D(z) = z + \mu(p)\bar{z}$. Hence, the Beltrami coefficient μ actually encode the conformality of f. Analyzing quasi-conformal f is equivalent to that for its associated Beltrami coefficient μ. To be detail, the angle of maximal magnification is $\arg(\mu(p))/2$ with magnifying factor $1 + |\mu(p)|$; for the maximal shrinking is the orthogonal angle $(\arg(\mu(p)) - \pi)/2$ with shirking factor $1 - |\mu(p)|$.

The maximal quasi-conformal dilation of f is given by

$$K = \frac{1 + \|\mu\|_\infty}{1 - \|\mu\|_\infty} \tag{3.4}$$

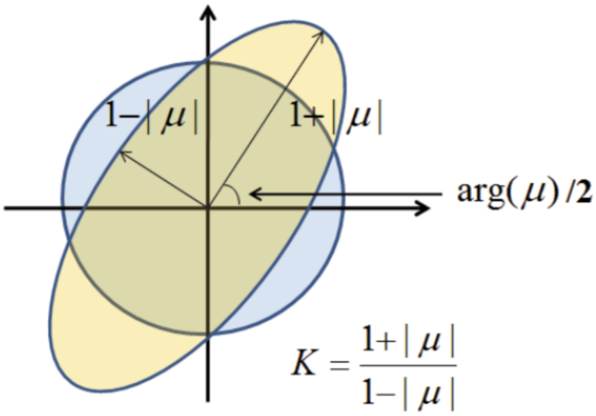

Figure 2: Illustration of how the Beltrami coefficient measures the conformality distortion of a quasi-conformal map

Figure 2 illustated the geometry of quasi-conformal map.

Another important relationship between a map and its Beltrami coefficients is the diffeomorphism property. By a norm constraint on μ, the bijectivity of f can be preserved which is explained as following theory.

Theorem 3.2. *If $f : \mathbb{C} \to \mathbb{C}$ is a C^1 map. Define*

$$\mu = \frac{\partial f}{\partial \bar{z}} / \frac{\partial f}{\partial z} \tag{3.5}$$

If μ satisfies $\|\mu_f\|_\infty < 1$, then f is bijective.

For two quasi-conformal maps f,g, the Beltrami coefficient of their composition can be expressed in terms of their individual Beltrami coefficients μ_f, μ_g directly according to the following theory.

Theorem 3.3. *For two quasi-conformal maps $f : \Omega \subset \mathbb{C} \to f(\Omega)$ and $g : f(\Omega) \to \mathbb{C}$, whose Beltrami coefficients are μ_f, μ_g respectively. The Beltrami coefficient of the composited function $g \circ f$ is clearly defined as*

$$\mu_{gof} = \frac{\mu_f + \left(\overline{f_z}/f_z\right)(\mu_g \circ f)}{1 + \left(\overline{f_z}/f_z\right)\overline{\mu_f}\,(\mu_g \circ f)} \tag{3.6}$$

Theorem 3.4. *Suppose $f : M_1 \to M_2$ and $g : M_2 \to M_3$ are two quasi-conformal maps. Write the associated Beltrami coefficient as $\mu_{f^{-1}}$ and μ_g respectively. Suppose $\mu_{f^{-1}} = \mu_g$. Then, the Beltrami coefficient of $g \circ f$ is equal to 0. In other words, $g \circ f : M_1 \to M_3$ is a conformal map.*

As we almost have everything for computing Beltrami coefficients from a given quasi-conformal map, we also need the converse. That's to say, given complex function μ, we can solve out it associated quasi-conformal map f if $\|\mu\|_\infty < 1$.

Given a Beltrami coefficient μ which is less than 1. Denote the corresponding quasi-conformal map $f : \mathbb{C} \to \mathbb{C}$ as $f = u + iv$, we have

$$\mu(f) = \frac{(u_x - v_y) + \sqrt{-1}\,(v_x + u_y)}{(u_x + v_y) + \sqrt{-1}\,(v_x - u_y)} \tag{3.7}$$

Rewrite $\mu = \rho + i\tau$. From the Beltrami Equation 3.1, we are able to use one pair of v_x, v_y or u_x, u_y to describe the partial derivatives of the other pairs with linear combinations as:

$$\begin{aligned} v_y &= \alpha_1 u_x + \alpha_2 u_y; \\ -v_x &= \alpha_2 u_x + \alpha_3 u_y \end{aligned} \quad \text{and} \quad \begin{aligned} -u_y &= \alpha_1 v_x + \alpha_2 v_y \\ u_x &= \alpha_2 v_x + \alpha_3 v_y \end{aligned} \tag{3.8}$$

where $\alpha_1 = \frac{(\rho-1)^2+\tau^2}{1-\rho^2-\tau^2}$; $\alpha_2 = -\frac{2\tau}{1-\rho^2-\tau^2}$; $\alpha_3 = \frac{(1+\rho)^2+\tau^2}{1-\rho^2-\tau^2}$. Since $\nabla \cdot \begin{pmatrix} -v_y \\ v_x \end{pmatrix} = 0$ and $\nabla \cdot \begin{pmatrix} -u_y \\ u_x \end{pmatrix} = 0$, we have

$$\nabla \cdot \left(A \begin{pmatrix} u_x \\ u_y \end{pmatrix} \right) = 0 \quad \text{and} \quad \nabla \cdot \left(A \begin{pmatrix} v_x \\ v_y \end{pmatrix} \right) = 0 \tag{3.9}$$

where $A = \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_2 & \alpha_3 \end{pmatrix}$ and is easily checked to be symmetric positive definite. Equation 3.9 is called the generalized *Laplace equation*. Solving the equation, so one can obtain everything for f. Note that here the information for α_1, α_2 and α_3 are given by the Beltrami coefficient μ. In a word, when we need to find an optimal quasi-conformal map, we can convert the problem to find an optimal complex-valued function alternatively.

Alternatively, quasi-conformal maps can also be defined between two Riemann surfaces $\mathcal{M}$ and $\mathcal{N}$ by *Beltrami differential* instead of Beltrami coefficient. A *Beltrami differential* $\mu(z)\frac{\overline{dz}}{dz}$ on $\mathcal{M}$ is an assignment to each chart (U_α, ϕ_α) of an L^∞ complex-valued function μ_α defined on the local parameter z_α such that

$$\mu_\alpha (z_\alpha) \frac{\overline{dz_\alpha}}{dz_\alpha} = \mu_\beta (z_\beta) \frac{\overline{dz_\beta}}{dz_\beta} \tag{3.10}$$

on the domain also covered by another chart (U_β, ψ_β), where $\frac{dz_\beta}{dz_\alpha} = \frac{d}{dz_\alpha}\phi_{\alpha\beta}$ and $\phi_{\alpha\beta} = \phi_\beta \circ \phi_\alpha^{-1}$. For a better illustration, we give figure 3.

3.2. Fourier Approximation for Beltrami Representation. [21] shows that Fourier approximation for Beltrami representation can easily preserve the mapping information while that for the representation of coordinate functions fails to enforce the homeomorphism. Compared with the latter method which requires that the Jacobian of the coordinate functions has to be greater than 0, the Fourier approximation for Beltrami representation requires only that the supreme norm must be less than 1, which is easier to be satisfied.

In the discrete case, an $N \times N$ Beltrami coefficient μ can be separated into two images representing the real and imaginary parts, μ_r and μ_i, respectively. The DFT of μ_r is

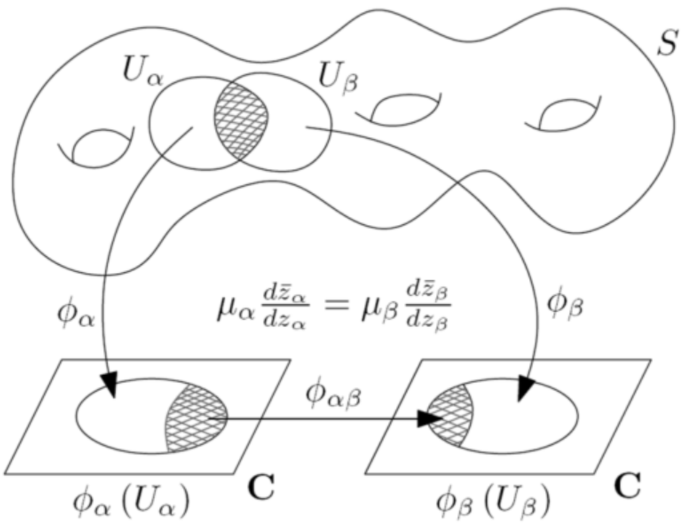

Figure 3: Beltrami differential on general Riemann surfaces

$$\hat{\mu_r}(m,n) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} \mu_r(k,l) e^{-\sqrt{-1}\frac{2\pi km}{N}} e^{-\sqrt{-1}\frac{2\pi ln}{N}} \tag{3.11}$$

This is equivalent to

$$\hat{\mu_r} = U\mu_r U \tag{3.12}$$

where $U_{kl} = \frac{1}{N}e^{-\sqrt{-1}\frac{2\pi kl}{N}}$, $0 \leq k,l \leq N-1$. The inverse DFT of $\hat{\mu_r}$ is

$$\mu_r(p,q) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} \hat{\mu_r}(m,n) e^{\sqrt{-1}\frac{2\pi pm}{N}} e^{\sqrt{-1}\frac{2\pi qn}{N}} \tag{3.13}$$

which can be rewritten as

$$\mu_r = (NU^*)\hat{\mu_r}(NU^*) \tag{3.14}$$

When using Fourier coefficients $\hat{\mu_r}, \hat{\mu_i}$ to approximate Beltrami coefficients, we keep only a small fraction of the frequency components which acts as the low-frequency components. This is equivalent to saying that only a small fraction of the frequency components can well capture the majority of deformation. Motivated by this idea, we propose the Beltrami Solver Network (BSNet) which takes a Beltrami coefficient as input and uses the Fourier approximation to represent the global information of the input.

3.3. Numerical Implementation of LBS. Given the Beltrami Coefficient μ, we can reconstruct the corresponding quasi-conformal mapping f by solving (3.9)

In discrete case, the parameter domain D is a mesh grid. The restriction of f on each triangular face T is linear and can be written as

$$f|_T(x,y) = \begin{bmatrix} u|_T(x,y) \\ v|_T(x,y) \end{bmatrix} = \begin{bmatrix} a_T x + b_T y + r_T \\ c_T x + d_T y + s_T \end{bmatrix} \tag{3.15}$$

Obviously, on each face, we have induced partial derivatives from the face-wise linear assumption.

Hence, the partial derivatives of f at each face T can be denoted as $D_x f(T) = a_T + ic_T$ and $D_y f(T) = b_T + id_T$. Now the gradient $\nabla_T f := (D_x f(T), D_y f(T))^t$ on T can be computed by solving

$$\begin{pmatrix} v_1 - v_0 \\ v_2 - v_0 \end{pmatrix} \nabla_T f_i = \begin{pmatrix} f_i(\vec{v_1}) - f_i(\vec{v_0}) \\ f_i(\vec{v_2}) - f_i(\vec{v_0}) \end{pmatrix} \tag{3.16}$$

where $[\vec{v_0}, \vec{v_1}]$ and $[\vec{v_0}, \vec{v_2}]$ are two edges on T.

Besides, μ is a face-based function. Denote the face-based function α_i on face T by α_i^T, where $i = 1, 2, 3$. From 3.8, we have

$$\begin{aligned} -d_T &= \alpha_1^T a_T + \alpha_2^T b_T \\ c_T &= \alpha_2^T a_T + \alpha_3^T b_T \end{aligned} \tag{3.17}$$

and

$$\begin{aligned} -b_T &= \alpha_1^T c_T + \alpha_2^T d_T \\ a_T &= \alpha_2^T c_T + \alpha_3^T d_T \end{aligned} \tag{3.18}$$

Let $T = [\vec{v_i}, \vec{v_j}, \vec{v_k}]$ and $\vec{w_I} = f(\vec{v_I})$, where $I = i, j, k$. Suppose $v_I = g_I + ih_I$ and $w_I = s_I + it_I$ $(I = i, j, k)$. Using (3.16), According to mapping (3.16), for each face T, we have

$$\begin{bmatrix} a_T & b_T \\ c_T & d_T \end{bmatrix} \begin{bmatrix} g_j - g_i & g_k - g_i \\ h_j - h_i & h_k - h_i \end{bmatrix} = \begin{bmatrix} s_j - s_i & s_k - s_i \\ t_j - t_i & t_k - t_i \end{bmatrix} \tag{3.19}$$

Thus,

$$\begin{bmatrix} a_T & b_T \\ c_T & d_T \end{bmatrix} = \frac{1}{2 \cdot Area(T)} \begin{bmatrix} s_j - s_i & s_k - s_i \\ t_j - t_i & t_k - t_i \end{bmatrix} \begin{bmatrix} h_k - h_i & g_i - g_k \\ h_i - h_j & g_j - g_i \end{bmatrix} \tag{3.20}$$

$$= \begin{bmatrix} A_i^T s_i + A_j^T s_j + A_k^T s_k & B_i^T s_i + B_j^T s_j + B_k^T s_k \\ A_i^T t_i + A_j^T t_j + A_k^T t_k & B_i^T t_i + B_j^T t_j + B_k^T t_k \end{bmatrix} \tag{3.21}$$

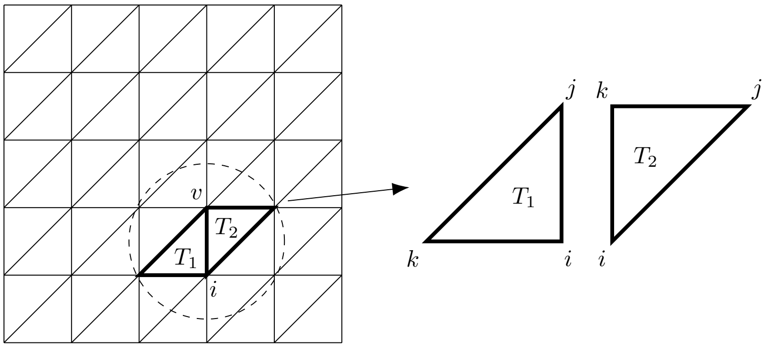

Figure 4: Illustration of the derivation of the coefficient of vertex v

where

$$
\begin{aligned}
A_i^T &= (h_j - h_k)/2 \cdot Area(T); \quad & B_i^T &= (g_k - g_j)/2 \cdot Area(T) \\
A_j^T &= (h_k - h_i)/2 \cdot Area(T); \quad & B_j^T &= (g_i - g_k)/2 \cdot Area(T) \\
A_k^T &= (h_i - h_j)/2 \cdot Area(T); \quad & B_k^T &= (g_j - g_i)/2 \cdot Area(T)
\end{aligned} \tag{3.22}
$$

For each vertex v_i, let N_i be the collection of neighborhood faces attached to v_i. By careful checking, one can observe that

$$\sum_{T \in N_i} A_i^T b_T = \sum_{T \in N_i} B_i^T a_T; \quad \sum_{T \in N_i} A_i^T d_T = \sum_{T \in N_i} B_i^T c_T; \tag{3.23}$$

Substituting Equations (3.17) and (3.18) into (3.23), we obtain the following equations

$$\sum_{T \in N_i} (A_i^T[\alpha_1^T a_T + \alpha_2^T b_T] + B_i^T[\alpha_2^T a_T + \alpha_3^T b_T]) = 0 \tag{3.24}$$

$$\sum_{T \in N_i} (A_i^T[\alpha_1^T c_T + \alpha_2^T d_T] + B_i^T[\alpha_2^T c_T + \alpha_3^T d_T]) = 0 \tag{3.25}$$

Replacing a_T, b_T, c_T and d_T with their corresponding expressions, we derive the following coefficient for the central vertex i of N_i

$$c_i = \sum_{T \in N_i} [\alpha_1^T (A_i^T)^2 + 2\alpha_2^T A_i^T B_i^T + \alpha_3^T (B_i^T)^2] \tag{3.26}$$

For two incident triangular faces $T_1, T_2 \in N_i$, the edge e between T_1 and T_2 connect the central vertex i and another vertex v, as shown in Figure 4. In this case, the index of vertex v in T_1 is j, and that in T_2 is k. The coefficient of vertex v can then be written as follow

$$c_v = \begin{array}{l} \alpha_1^{T_1} A_i^{T_1} A_j^{T_1} + \alpha_2^{T_1}(A_i^{T_1} B_j^{T_1} + A_j^{T_1} B_i^{T_1}) + \alpha_3^{T_1} B_i^{T_1} B_j^{T_1} + \\ \alpha_1^{T_2} A_i^{T_2} A_k^{T_2} + \alpha_2^{T_2}(A_i^{T_2} B_k^{T_2} + A_k^{T_2} B_i^{T_2}) + \alpha_3^{T_2} B_i^{T_2} B_k^{T_2} \end{array} \tag{3.27}$$

According to Equation (3.26), (3.27), (3.24) and (3.25), for a vertex i, we can write down the following equations

$$\begin{array}{rcl} c_i s_i + \sum_{v \in V_i} c_v s_v & = & 0 \\ c_i t_i + \sum_{v \in V_i} c_v t_v & = & 0 \end{array} \tag{3.28}$$

where V_i is the set of adjacent vertices of vertex i.

For an $N \times N$ mesh grid, we have

$$\begin{array}{rcl} C_s \boldsymbol{s} & = & \mathbf{0} \\ C_t \boldsymbol{t} & = & \mathbf{0} \end{array} \tag{3.29}$$

where $\boldsymbol{s}$ and $\boldsymbol{t}$ are the $N \times N$ dimensional coordinate vectors, in which the entries of boundary constraints set to their true values. C_s and C_t are the same $N^2 \times N^2$ sparse matrix, each row of which contains c_i and c_v for a vertex in the mesh grid. The only difference between C_s and C_t is that the rows that correspond to the boundary constraints of the two coordinates are set to 0.

Solving this linear system with the boundary constraints, we can obtain the corresponding mapping f given a Beltrami coefficient μ_f.

3.4. Convolutional Neural Network. Deep learning is a branch of Machine Learning. With the development of computation power, deep learning become popular and helped multiple fields like computer vision, natural language processing, financial modeling, etc. Imitating the way human beings learn from experiences, supervised deep learning methods learn to solve a task by doing regression from data. A deep learning model, which is generally a neural network, consists of three main parts: architecture, optimization method, and the loss function.

For the architecture, the neural network is made of neurons that mimic the functionality of human brain. For each neuron, the main components include the input feature $x_1, x_2, \ldots, x_n$, thier correspondending weighting $\omega_1, \omega_2, \ldots, \omega_n$, the transfer function which is simply summation in most cases, the bias b and the activation function ϕ. When a group of feature passes through this neuron, it will output a signal y, which is conputed by

$$y = \phi\left(\sum_{i=1}^{n} x_i \omega_i + b\right) \tag{3.30}$$

The output y of this neuron will be the input of the neurons connected to this one in the after layer. Through such a process, the signals are transmitted layer by layer, and features are processed to produce the final output. In recent years, the neural network become deeper and deeper. This may be explained as more layers would enrich the levels of features and

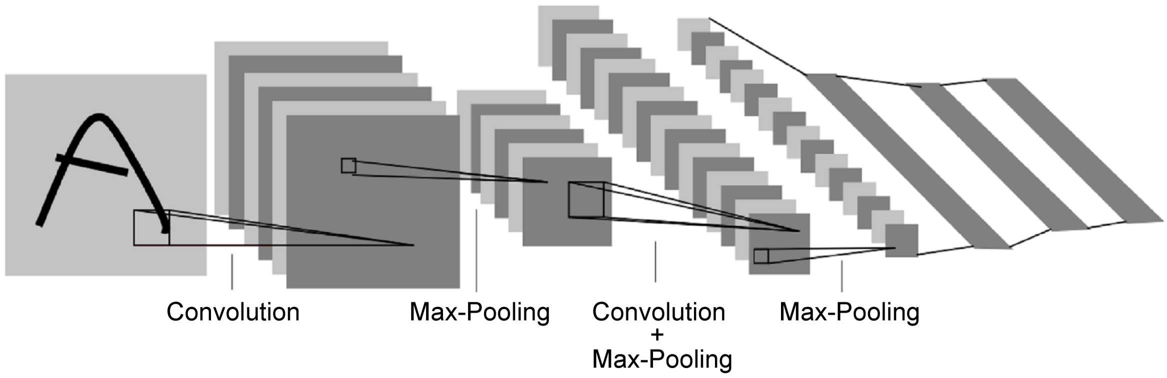

Figure 5: Convolution Neural Network

leads to a better result. However, a network model with too many layers is hard to train and get to convergence. Through deep layers can bring more non-linearity and flexibility to approximate the function that describes the problem to be solved, too many parameters may result in vanishing and exploding gradient problems [10]. For tasks like image analysis, pixel signals are defined as associated with their neighbors. In other words, to extract the features like edges, shapes, textures, and objects, one should not only consider this pixel only but also should account for its relation with others nearby. Besides such a need for consideration on local, the location of some objects may be different spatially in the image domain. The convolutional neural network, which is made of layers of trainable filters, is to solve such tasks. In a convolution neural network, the neuron's input feature is encircled by a moving window while the output will be located at the place accordingly (see Figure.5).

During the training of a deep neural network model, the data are input to generate predictions. To evaluate how well the model is trained, we need to compare the difference between the prediction and the ground truth. Such a difference is not always the Euclidean distance between them. For example, for a classification task, cross-entropy loss is used more often due to its capacity on measuring the disorder and unpredictability of a system. When we design the model to output a probability distribution $q(x)$ where the input should belong, using cross-entropy loss can enhance the data cluster into groups and approach the actual distribution $p(x)$.

$$E(p,q) = -\sum p(x)\log(q(x)) \tag{3.31}$$

After the loss function and architecture are defined, the distance in the output space and the function to be regressed are fixed. The problem that remained is to optimize the whole model to obtain a small difference between the predicted and the target. To reduce the difference between the output of the network and the ground truth, the weights ω_i for each neuron k should be updated by gradient descent. For the error defined according to some loss function $E(Y, \hat{Y})$ where Y is the label and the prediction $\hat{Y} = N(X; \Omega)$, where N is the neural network with weight parameters Ω.

The gradient for each ω is computed based on the chain-rule. This greatly reduce the

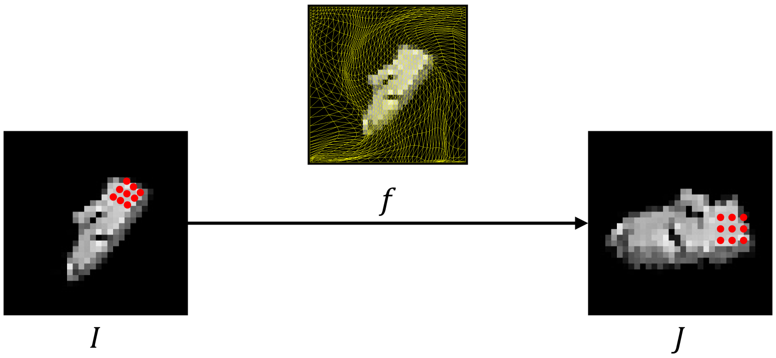

Figure 6: Illustration for deformable convolution via deformable feature map: do regular convolution on the deformed feature map $J = I \circ f$ is equvalent to do deformable convolution on the original feature map I

descent computation between layers. To make it clear, suppose the we have $N + 1$ layers in total, number the first(input layer) as 0 and the last(output layer) as N. Denote k-th layer has M_k nodes. The weight parameter from node i in $m - 1$ layer to node j in m layer is ω_{ij}^m. The The descent for weights from the last hidden layer $N - 1$ to the output layer N is

$$\frac{dError}{d\omega_{ij}^m} = \frac{dError}{dx_j^m}\frac{dx_j^m}{d\omega_{ij}^m} \quad \text{for } i = 1, \ldots, M_{m-1}; j = 1, \ldots, M_m \tag{3.32}$$

where $x_j^m = h_j^m\left(\sum_{i=1}^{M_{m-1}} x_i^{m-1}\omega_{ij}^m\right)$ is the activated neuron value for node j in the m-th layer.

Then descent for middle hidden layer k is

$$\frac{dError}{d\omega_{ij}^k} = \frac{dError}{dx_j^k}\frac{dx_j^k}{d\omega_{ij}^k} \quad \text{for } i = 1, \ldots, M_{k-1}; j = 1, \ldots, M_k \tag{3.33}$$

where $x_j^k = h_j^k\left(\sum_{i=1}^{M_{k-1}} x_i^{k-1}\omega_{ij}^k\right)$ is the activated neuron value for node j in the k-th layer.

4. Methodology.

In this section, we will introduce our *Quasi-Conformal transformer*, which can deform the feature maps through a mapping learned from training data. To make it convenient, let's assume our feature map is a one-channel feature map, which is $I : \mathbb{R}^{H\times W} \longrightarrow \mathbb{R}$, where H, W denotes the height and width respectively. For multi-channel images, we can do the same as that for one channel for each channel of it. The pipeline is like this: 1) Input a feature map I into *Beltrami Generator*, which is a simple and small network, to obtain the Beltrami coefficient μ of a Quasi-conformal mapping. 2) Input the Beltrami coefficient μ into a pre-trained *Beltrami Solver Network* (BSnet) to acquire the associated Quasi-conformal mapping f_μ. 3) Spatially transform the feature map I with mapping f_μ as $I \circ f_\mu$. 3) Input the deformed image $I \circ f_\mu$ to a classifier or segmentation network according to the particular task. The architecture for our Quasi-Conformal transformer network is illustrated as Figure.7.

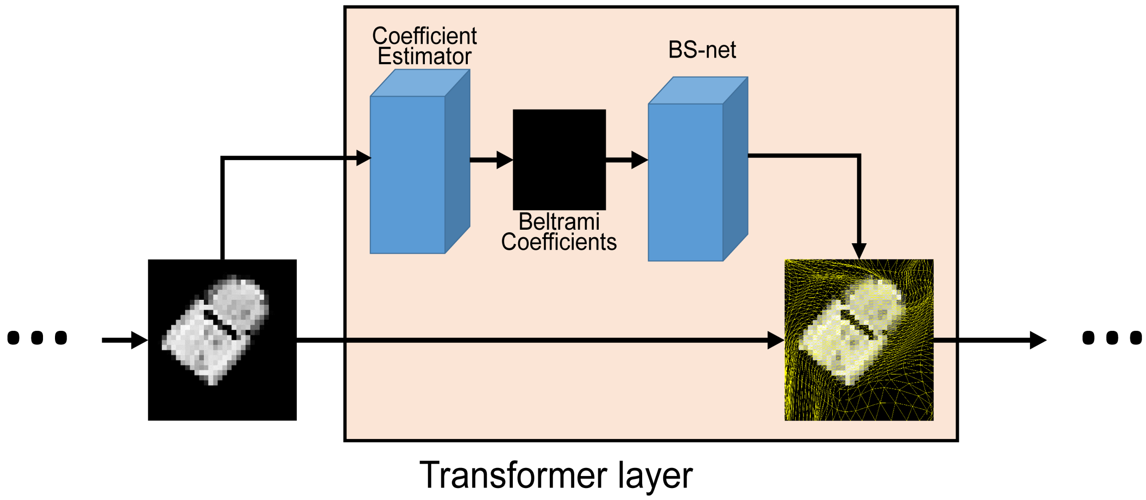

Figure 7: Quasi-Conformal transformer network

The whole model is an end-to-end neural network model without manual interference. Except for the BS-net, which is to solve for the quasi-conformal mapping given the Beltrami coefficients and is fixed as long as it's pre-trained, every parameter in the network is learnable. In the following, we will introduce the grid parameterization of images, the Beltrami coefficient generator, Beltrami solver network, and spatial transformation in detail. Besides these, we will discuss some training tricks for the overall Quasi-conformal transformer network at the end of this section.

4.1. Beltrami Solver Network. To make the ideal feasible, the first component required is a network to convert Beltrami coefficients to their corresponding mappings which is proposed by [1]. In the previous work, [21], the conversion from Beltrami coefficients to mappings is achieved by using Linear Beltrami Solver (LBS), in which a sparse linear system is required to be solved. In this paper, we are going to use a neural network to approximate the mappings given their corresponding Beltrami Coefficients. There are two benefits. On the one hand, once trained, a well-designed neural network gives predictions much faster than solving the sparse linear system. On the other hand, the neural network can backpropagate errors from its output to input, which makes it possible to use a trained neural network as a component when training another network to solve a complicated task.

From above we notice that a single Beltrami coefficient μ_{ij} represents the distortion of a local region, where i and j represent the indices of a triangle in the spatial position. The global distortion depends on the entire Beltrami coefficient μ, since the mapping is obtained by solving the linear system. It is natural that we can use cascaded convolutional and down-sampling layers to extract global information from the Beltrami coefficients, which can then be used to predict the mappings. The network structure should be similar to U-Net.

However, we learn the prior knowledge from [21] that Beltrami representation can be easily compressed by Fourier approximation, which means that low-frequency components hold most of the global distortion information. With this prior knowledge, we can further simplify the network structure. Compared with the convolutional layer, Fourier Transform has no trainable parameter and is much faster. And the generated low-frequency component has only two channels, much less than the deep features extracted by neural networks. As a result, we use Fourier Transform to extract global information.

Once we use the Fourier approximation, we have to think about how to combine spatial operations, such as convolution and interpolation, and the Fourier coefficients on the frequency domain. Directly performing such operations on the frequency domain easily leads to poor performance. It is critical to introduce a layer for transforming the frequency features to the spatial domain.

Considering that the size of the deep spatial features or Fourier coefficient mentioned above is different from the input images and the Beltrami coefficient, we cannot use (3.12) and (3.14) directly.

In order to tackle this problem, we propose the Domain Transform Layer (DTL) which imitates the computation of (3.12) and (3.14). This layer can be formulated

$$
\begin{aligned}
\hat{\mu} &= M\mu N = (\mu^T M^T)^T N \\
\mu &= M\hat{\mu} N = (\hat{\mu}^T M^T)^T N
\end{aligned}
\tag{4.1}
$$

where M and N are trainable complex matrices. This layer can be implemented by stacking two 1×1 convolution layers, together with some permutation operations.

As shown in subsection 4.1, given an input feature map of shape (H, W, C), we can permute the feature map to be (H, C, W). Then we perform K kernels 1×1 convolution. The shape of the resulting feature map should be (H, C, K). These operations are equivalent to matrix multiplication of a $H \times W$ matrix and a $W \times K$ matrix.

From (4.1) we notice that (4.1) can be implemented by stacking two matrix multiplication blocks. The detail structure is shown in Figure 8. In our experiments, $H = W = K = L = 14$.

With DTL, the features in the spatial domain can be transformed to a proper domain where spatial operations work. We can then perform convolution and upsampling on the features to obtain the mappings.

However, during the experiments, we found that the output mappings of the network are quite similar to their ground truth. But the outputs of the network have fewer details. In order to remedy this problem, we introduce a second path to improve the local details of the distortion, which are discarded in the computation of approximated Beltrami representation in the first path.

As in Figure 10, the second path is the upper path. In this skip path, convolution and a downsampling are performed on the input μ, after which two more convolution layers are performed and the output features are concatenated to the output from the first path. Experimental results in the following sections show the necessity of this skip path.

We trained this network in an unsupervised setting. As mentioned in subsection 4.1, a single value in the Beltrami coefficient represents the distortion of a triangular face. The shape of the Beltrami coefficient describing an $N \times N$ image should be $(N-1) \times [2 \times (N-1)]$. Due to this reason, LBS takes an $(N-1) \times 2 \times (N-1)$ dimensional vector as its input. However, in this paper, we wish to ensure the consistency between the input and output of our model. To cope with this conflict, we adopt the following method.

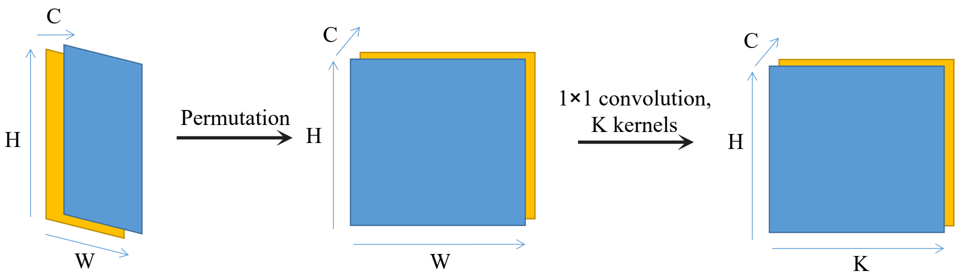

Figure 8: Matrix multiplication in the domain transform layer

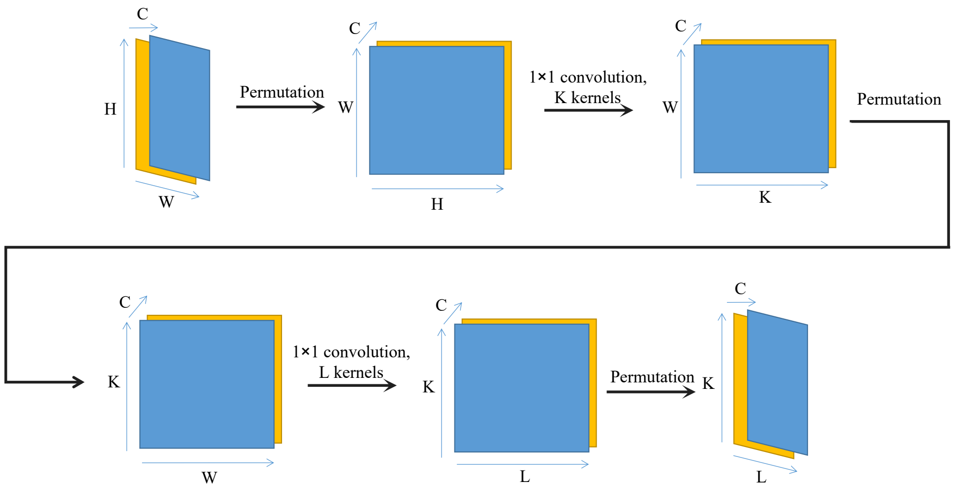

Figure 9: Domain Transform Layer

Countless $N \times N$ Beltrami coefficients μ_{sqr} are generated by stacking pairs of images in the ILSVRC2012 dataset, which are augmented with some data augmentation tricks, such as random crop and flipping. μ_{sqr} serves as the input of BSNet, representing the Beltrami coefficients of the triangles with odd indices in each row. An $(N-1) \times [2 \times (N-1)]$ Beltrami coefficient μ_{rect} is then obtained by removing the last row and column of μ_{sqr} and interpolating the Beltrami coefficients representing the triangles with even indices with the values representing the surrounding faces. the linear system in the LBS can then be retrieved with μ_{rect}, the coefficients of which are then used to compute the loss.

In Quasi-conformal geometry, every vertex with index i in a mesh satisfies Equations (3.29). It is natural to regard Equations (3.29) as a loss function when training a neural network to approximate the ground truth mapping of a given Beltrami coefficient μ.

The loss function can be formulated as follow

$$L_{Lapla} = \frac{1}{2N^2}(\|C_s \boldsymbol{s}\|_1 + \|C_t \boldsymbol{t}\|_1) \tag{4.2}$$

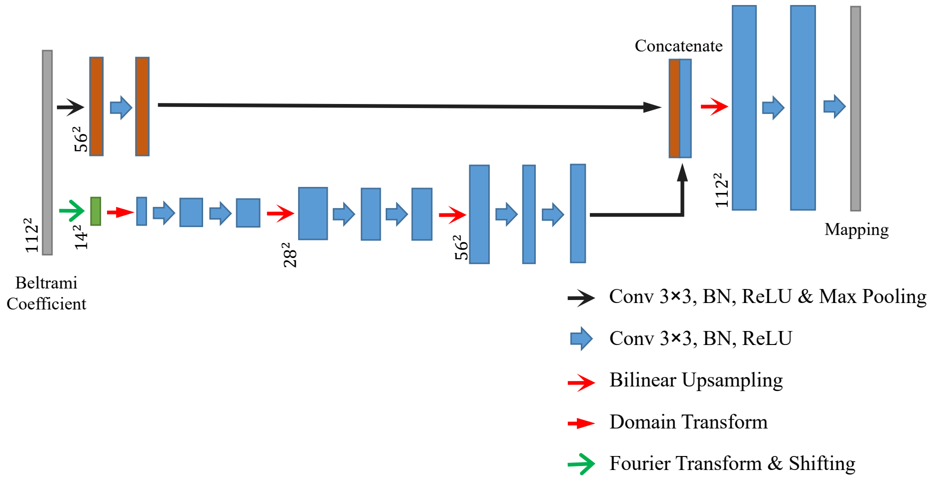

Figure 10: The architecture of the Beltrami Solver Network (BSNet).

$$L_{BSNet} = \gamma L_{Lapla} \tag{4.3}$$

Note that each row in C_s and C_t represents the relationship between a certain pixel and the pixels adjacent to it. In the context of a triangular mesh, a pixel is adjacent to at most six pixels. So each row in C_s and C_t has at most seven nonzero elements. Although the two $N^2 \times N^2$ matrices C_s and C_t are sparse, both matrices can be rewritten as two dense arrays in the implementation of our method, which means that the computation of L_{BSNet} is memory saving and efficient.

4.2. Beltrami Coefficient Estimator. Another component in the proposed algorithm is the Estimator Network which takes the source and target images as its input and generates the corresponding Beltrami coefficient representing the distortion of the input images. The network is based on U-Net with an activation added to its output. Its framework is shown in Figure 11.

The estimator performs convolution and pooling to extract the deep spatial features of the two input images. These deep spatial features are then up-sampled to be a two-channel image $\tilde{\mu}$ from which the norm and angle of $\tilde{\mu}$ can be computed. Notice that when we train Estimator, we assume that a BSNet has been trained so that for any μ satisfies $\|\mu\|_2 < 1$, it can convert the μ to its corresponding mapping. Naturally, μ generated by Estimator should also satisfy this condition. In order to ensure $\|\mu\|_2 < 1$, we add a $Tanh$ activation which takes $\tilde{\mu}$ as its input.

$$\|\mu\|_2 = Tanh(\|\tilde{\mu}\|_2)$$

where

$$Tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

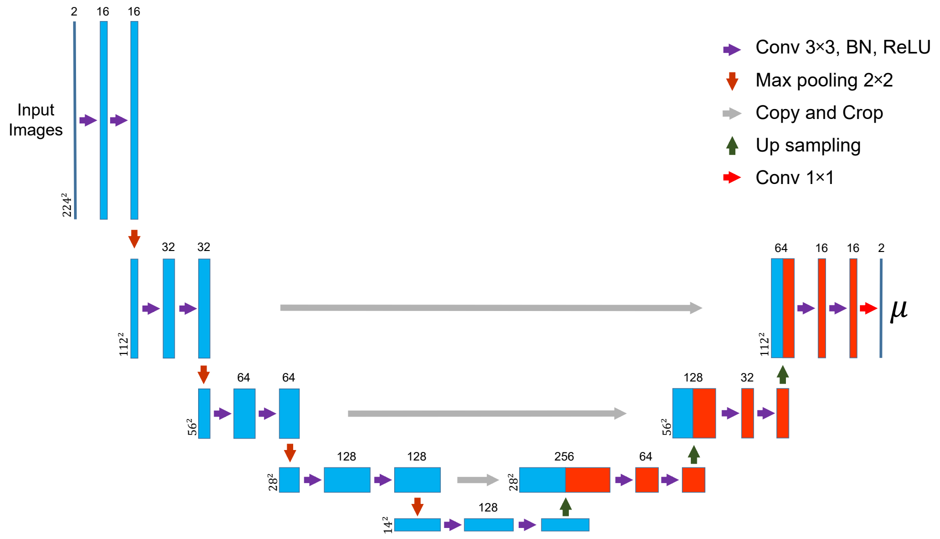

Figure 11: The architecture of the Estimator Network.

Since $\|\tilde{\mu}\|_2 \geq 0$, $1 > \|\mu\|_2 \geq 0$ always holds. Then we have

$$Re(\mu) = \|\mu\|_2 cos(Arg(\tilde{\mu})) \\ Im(\mu) = \|\mu\|_2 sin(Arg(\tilde{\mu})) \tag{4.4}$$

μ serves as the Beltrami coefficient representing the deformation between the two input images.

We also use the following loss function to suppress the norm of μ

$$L_\mu = \frac{1}{N}\sum_{n=1}^{N} \|\mu\|_2^2. \tag{4.5}$$

The smoothness can be enhanced by

$$L_{smooth} = \frac{1}{N}\sum_{n=1}^{N} \|\nabla\mu\|_2^2. \tag{4.6}$$

The fidelity term is as follow

$$L_F = \frac{1}{N}\sum_{n=1}^{N} \|I_s \circ f - I_t\|_2^2. \tag{4.7}$$

The total loss function is

$$L_{Estimator} = \alpha L_F + \beta L_\mu + \eta L_{smooth} \tag{4.8}$$

where α, β, and η are hyper-parameters that give different weighting to the corresponding terms.

4.3. Grid Parameterization and Spatial Transformation. To make it convenient to perform the pixel-wise transformation on images, we need to parametrize the image domain with coordinates systems. Notice that overall in this paper, the pixel would be referred to not only for elements in images but also for feature maps in the hidden layers of a neural network model. Also, we don't necessarily distinguish between image and feature map in this work. For a image with a height of H and width of W, to parameterize it within the domain $[0,1]^2$, we discretize the whole image domain by a grid $G = \{(x_i, y_j)\} = \{(ih_x, jh_y) : i = 1, 2 \ldots, H; j = 1, 2 \ldots, W\}$, where $h_x = \frac{1}{H+1}$ and $h_y = \frac{1}{W+1}$. In case the feature map contains multiple channels, the pixels for each channel located at the same position are parameterized by the same point in the grid.

In our implementation, the target image will always be parametrized with such a regular and normalized grid defined above. Then what we need to deform the image is the source coordinates on the input image. More cumulatively speaking, for the target coordinate (x_i^t, y_j^t), its corresponding pixel is the point whose pixel coordinate is (x_i^s, y_j^s) and lie in the input feature map. To summary, the output of the coefficient estimator and BSnet should be the coordinates of the source grid $G_s = \{(x_i^s, y_j^s)\}$ which sampled on the input image.

Through such a setting, we can simply use the embedding function *grid_sample()* in *PyTorch*, which is also the standard parameterization method used in spatial transformer network[11] and texture mapping in computer graphcis [7].

When the source coordinates are determined, generally its not exactly the same to any of the coordinates that prameterized the input image. Thus, interpolation for the resampling on points of source coordinates is must. The method for this interpolation should be differentiable to enable back-propogation. Associated with an input feature map I defined on $(x, y) \in \{(x_p, y_l) : p = 1, \ldots, H'; l = 1, \ldots, W'\}$ and the output mapping of coefficient estimator and BSnet $f : (x^t, y^t) \longrightarrow (x^s, y^s)$, we do resampling for $J = I \circ f$ which is defined on $(x^s, y^s) \in \{(x_i^t, y_j^t) : i = 1, \ldots, H; j = 1, \ldots, W\}$. We can write the resampling as:

$$\begin{aligned} J(x_i^t, y_j^t) = \sum_{p=1}^{H'} \sum_{l=1}^{W'} I(x_p, y_l) k(x_i^s - x_p; \Phi_x) k(y_j^s - y_l; \Phi_y) \\ \text{where } i = 1, \ldots, H \text{ and } j = 1, \ldots, W \end{aligned} \tag{4.9}$$

where Φ_x and Φ_y are the parameters of a generic sampling kernel $k()$, $G_0 = \{(x_p, y_l)\}$ is a regular and normalized grid that parameterized the image domain of I. The interpolation would be identical for each channel of the image. Let's fix the resampling method as bilinear, then the resampling method can be written as

$$J(x_i^t, y_j^t) = \sum_{p=1}^{H'} \sum_{l=1}^{W'} I(x_p, y_l) \max(0, 1 - \frac{|x_i^s - x_p|}{h_x}) \max(0, 1 - \frac{|y_j^s - y_l|}{h_y}) \quad \text{where } i = 1, \dots, H \text{ and } j = 1, \dots, W \tag{4.10}$$

Thus, the derivative of the bilinear interpolation for output feature map J at (x_i^t, y_j^t) with respect to the input feature map and the source coordinates are given respectively by

$$\frac{\partial J}{\partial I(x_p, y_l)}(x_i^s, y_j^s) = \sum_{p=1}^{H'} \sum_{l=1}^{W'} \max(0, 1 - \frac{|x_i^s - x_p|}{h_x}) \max(0, 1 - \frac{|y_j^s - y_l|}{h_y}) \tag{4.11}$$

$$\frac{\partial J}{\partial x^s}(x_i^s, y_j^s) = \sum_{p=1}^{H'} \sum_{l=1}^{W'} I(x_p, y_l) \max(0, 1 - \frac{|y_j^s - y_l|}{h_y}) h(x_i^s) \tag{4.12}$$

where

$$h(x_i^s) = \begin{cases} 0 & \text{if } |x_p - x_i^s| \geq h_x \\ \frac{1}{h_x} & \text{if } x_p \geq x_i^s > x_p - h_x \\ -\frac{1}{h_x} & \text{if } x_p < x_i^s < x_p + h_x \end{cases} \tag{4.13}$$

and similarly to 4.12 and 4.13 for $\frac{\partial J}{\partial y^s}$.

Through such a differentiable resampling method, the gradients in the neural network model can pass backward and enable the updating of the parameters. With the derivative $\frac{\partial J}{\partial x^s}$ and $\frac{\partial J}{\partial y^s}$, the transformer layer can be trained. Through the gradients with respect to the input feature map in Equation.4.11, the gradients are able to pass to the previous layer that outputs it and leads to an end-to-end trainable neural network model.

4.4. Quasi-Conformal Transformer Network. The combination of the Coefficient Estimator, Beltrami Solver network, and image warper forms up the Quasi-conformal transformer layer(see Figure. 7). Coefficient Estimator takes the input feature map I and predicts the Beltrami coefficients μ that is the quasi-conformal representation for the desired mapping f. The predicted Beltrami coefficients are solved into the mapping by the pre-trained BS-net. Directly by the mapping which is the source coordinates $(x^s, y^s) = f(x^t, y^t)$ that lie on the input feature map, we use image warper to do resampling and obtain the transformed map. In this model, except for the Beltrami Solver Network, which should remain static after it acquired enough pretraining, every module in the quasi-conformal transformer layer is trainable. This enables a quasi-conformal transformer to be inserted in any position of the convolutional neural network and ends up with an end-to-end trainable neural network model.

The operation of what a quasi-conformal transformer layer did in a network is explicit and can be interpreted visually. Like the idea of spatial transformer network [11], our quasi-conformal transformer network can assign the input feature map a trainable warping that can help the after layers work better and minimize the overall loss function. For example, in a classification task with disturbing images, a transformer layer can be inserted before the classifier network to restore the disturbance and recover the semantic meaning for a better classification result. Simply adding it to some specific tasks can also make sense. Like some popular works aim for learning convolutions that differ from the regular rectangle windows[6][34], our quasi-conformal transformer can also learn deformable convolution definition by warping the feature map into a deformed one. In this way, doing regular convolution in the deformed map is mathematically equivalent to assigning deformable convolution in the original feature map.

Beyond what a spatial neural network can do, quasi-conformal transformers are highlighted to learn dense and large transformations that are calculated point-wisely for the input feature map. Though it would also be possible for a thin-plate spline (TPS) variant of the spatial neural network, that directly predicts the source coordinates, our method can be much easier to control the topology and the degree of the deformation benefit the Beltrami representation. Controlling such topologic and geometric properties of a spatial transformation is very important in learning a large and dense deformation. To be in more detail, learning a mapping that is too flexible and without proper constraint may easily result in overfitting. The loss can converge to a very low point easily due to the flexibility of the mapping but failed to do prediction correctly in the testing. This will be evaluated thoroughly in Section. 5.3.

To achieve this, we carefully designed a penalty term that can constrain the mapping to be a diffeomorphism and enhance its property of topology-preserving. Denote the quasi-conformal transformer layer as N_{qct} and the other parts of the network as V, such a penalty term is written as:

$$\begin{aligned} E_{reg}(V, N_{qct}) &= \int |\mu| + \int |\nabla \mu| \\ &= \sum_{i=1}^{H}\sum_{j=1}^{W} \mu(x_i^t, y_j^t) + \sum_{i=1}^{H}\sum_{j=1}^{W} \nabla\mu(x_i^t, y_j^t) \end{aligned} \tag{4.14}$$

The penalty term should be put in the final loss function. Thus, for a particular task, the overall cost function is:

$$E = E_{task} + \alpha E_{reg} \tag{4.15}$$

where E_{reg} is defined as Equation.4.14 weighted by α, while E_{task} is the penalty term associated with the task, e.g. cross-entropy error for classification or mean square error for registration.

In the work of this paper, the

5. Experimental Results. In this section, we will thoroughly evaluate the capability of the quasi-conformal transformer network and compare it with its most related existing work[11] and its thin-plate spline-based variant mentioned in their work. Firstly, like the work of spatial transformer, we test the power for localization on distorted versions of MNIST[19] handwriting dataset. In this experiment, the images will be transformed affinely. Such deformed images will only have characters on a small region of the image domains which makes it challenging to do classification. Through our quasi-conformal transformer layer, the convolution will only be performed on regions that contain information. The second experiment is to evaluate the elastic deformation restoration of the quasi-conformal transformer. To do this, we randomly deform the images from CIFAR10[14] and employ such deformed CIFAR10 to do supervised classification. We compare between two networks, one is a simple CNN classifier network without a quasi-conformal layer while another is the same classifier with the quasi-conformal transformer layer in its head. Our QCTN can restore the deformed image and result in better classification accuracy. In the third experiment, we will show experiments on Fashion MNIST[31] data that can do localization and elastic deformation restoration simultaneously. Lastly, on the original CIFAR10 dataset without any pre-processing, we put a QC transformer layer ahead of a classifier and do training without pretraining on the QC transformer layer. From the deformation by QC transformer on the feature map, we are able to do a learnable deformable convolution on images. The results promise quasi-conformal layers helped to improve the classification accuracy.

Our method was implemented in *Python* and run on centOS-7 based central cluster nodes with a 2.4GHz Intel Xeon E5-2680 CPU, 64GB, and a GeForce GTX 1080 Ti GPU. The learning rate in the stage of classification is 0.00005. What is worthy to mention is, when we determine the parameter for learning rate, it's hard for a thin-plate spline spatial transformer network to converge. A parameter explosion and overfitting can easily occur during its training. However, for a fair comparison, we take the learning rate to be 0.00005 uniformly in this work and do an extra clip for gradients especially when implementing the TPS variant for STN-CNN by 10.

5.1. Localization on Deformed MNIST. We take MNIST handwriting dataset to illustrate the ability of the quasi-conformal transformer network for localizing the characters that are coarsely standing only in a partial region of the whole image domain. We first affinely deform the MNIST images. The deformation range is set to enable $[-\frac{\pi}{3}, \frac{\pi}{3}]$ for rotation angle θ and $[0.2, 0.6]$ for scaling parameter s. The translation is allowed as long as the characters will not move out of the image domains. We didn't test that for elastic deformation with MNIST since the elastic deformation on characters is not visually obvious. However, we do test that for elastic deformation with Fashion MNIST and will be presented in the later section. We trained three networks in this part including our quasi-conformal network to compare the performance. The other two models include the baseline convolutional neural network, which contains three convolutional layers and two fully-connected layers and the same network with spatial transformer (ST-CNN) or quasi-conformal transformer (QCT-CNN) layer inserted between the input and the classifier CNN. The spatial transformer and quasi-conformal transformer layer are both pretrained to obtain an initialization. In the process of pretraining, the transformer layer takes the deformed as input and the original image as the label to update the weightings in the module.

In the stage for the classification where the overall modal is trained together, the optimization will run for 100 epochs. We evaluate the classification accuracy rate on the test dataset. Compared to the baseline convolutional neural network, both models equipped with transformer acquire better results as the accuracy rate given in Table. 1. From the visualized results shown in Figure. 12, it's obvious that the transformer succeeds in conferring the feature map and focuses only on regions that encircle the characters. However, our methods

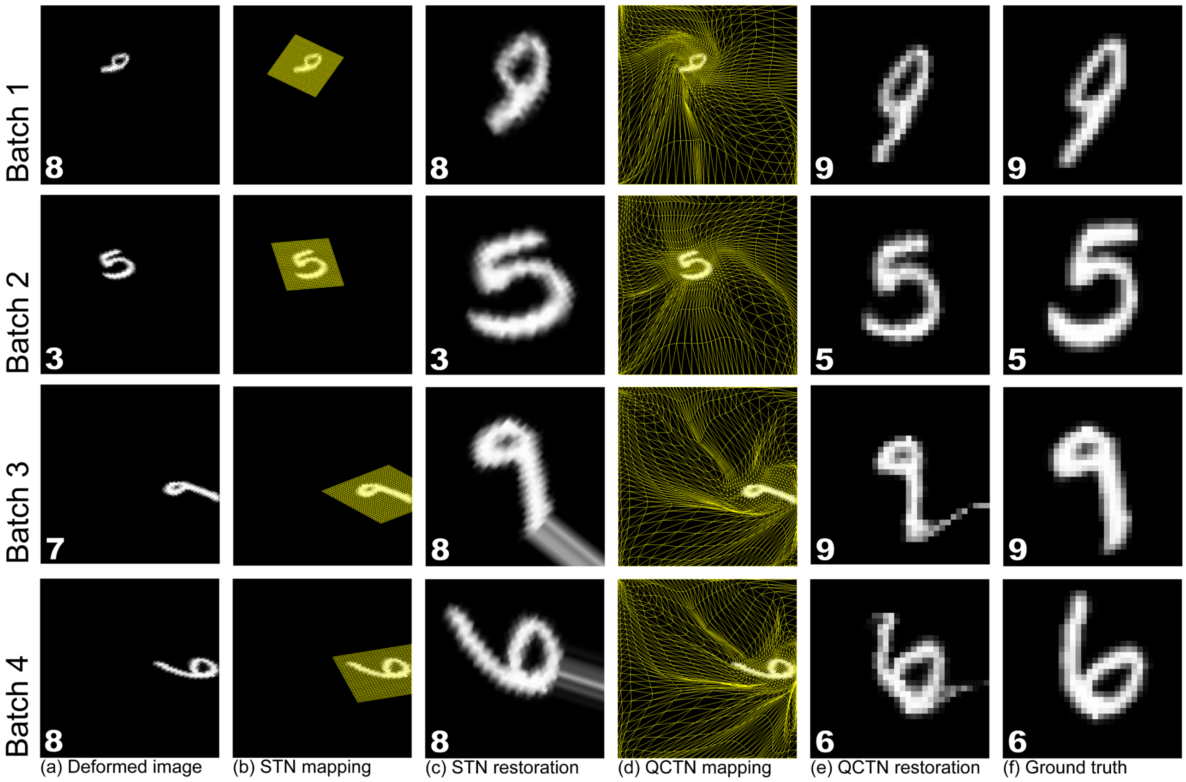

Figure 12: Classification result on affinely deformed MNIST. (a) the deformed images. (b) mapping generated by STN visualized on deformed image. (c) image localized by STN. (d) mapping generated by QCTN visualized on deformed image. (e) image localized by QCTN. (f) ground truth. The characters on the down-left of column (a)(c)(e) indicates the predicted result by baseline CNN, STN and QCTN respectively.

Table 1: The results for classifying affinely transformed MNIST by baseline CNN, STN and QCTN

Method	Train	Test
CNN	83.62	82.73
ST-CNN	94.97	94.90
QCT-CNN	96.45	96.32

outperform STN not only by quantity but also by the quality of images transformed. With point-wise transformation by our QC transformer, the characters are more clean and sharp with little blur. Particularly, when the character lies close to the boundary of the domain, STN will generate a large blur because of bilinear interpolation on boundaries. However, our method, which is a point-wise learnable mapping, can reduce such advantages efficiently.

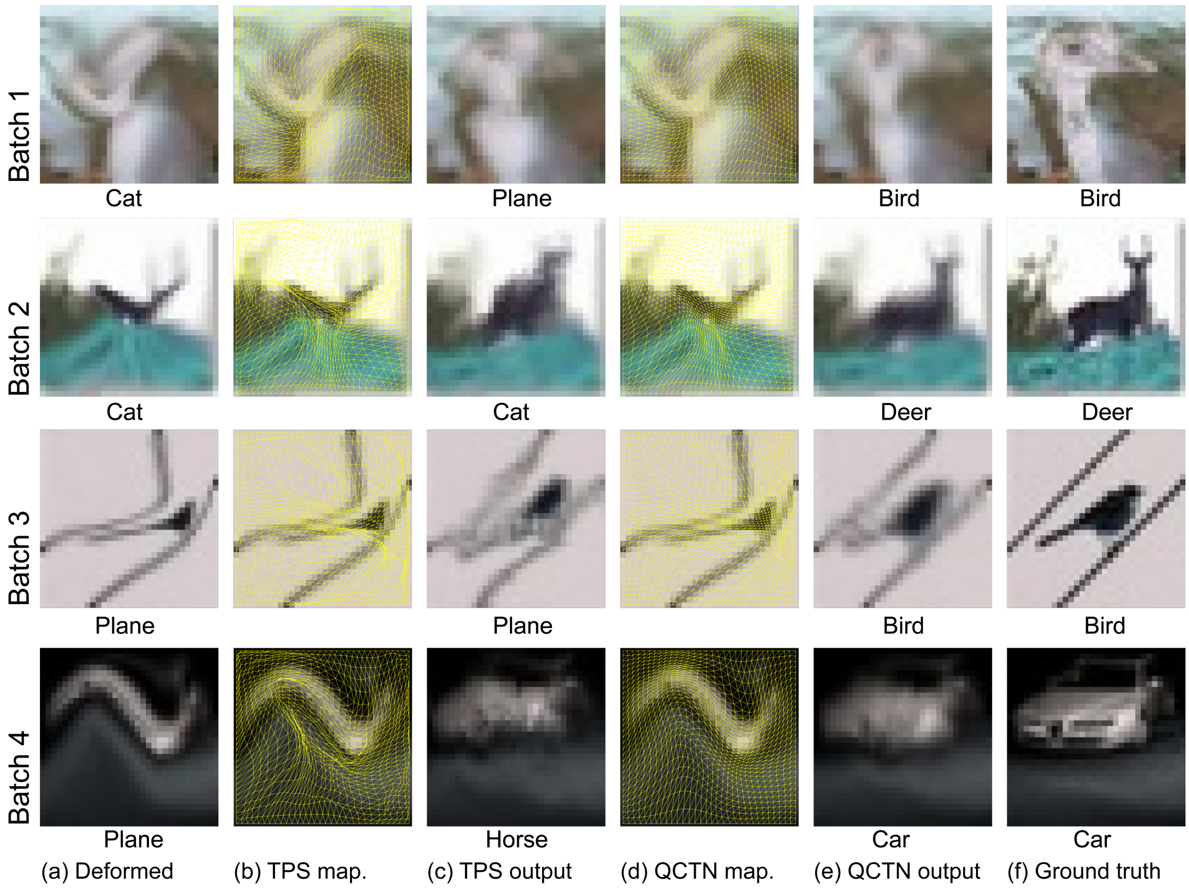

Figure 13: Classification result on CIFAR-10 with large non-rigid deformation. (a) the deformed images. (b) mapping generated by TPS-STN visualized on deformed image. (c) image recovered by TPS-STN. (d) mapping generated by QCTN visualized on deformed image. (e) image recovered by QCTN. (f) ground truth. The class names on the bottom of images in column (a)(c)(e) indicates the predicted class by baseline CNN, TPS-STN and QCTN respectively.

5.2. Elastic Deformed CIFAR-10. In this section, we perform an experiment to show that the quasi-conformal transformer can learn to be invariant to elastic deformation. That's to say, given a distorted image that may come from capturing across some uneven surfaces like glasses or water, the quasi-conformal transformer layer can restore the image before it goes into the classification network. Through such restoration, the original semantic meaning of the image should be recovered. We assign the elastic deformation with different scales on the dataset CIFAR10, which is a small low-resolution image recognition dataset, to obtain the deformed dataset. For comparison, we also implemented a variant of spatial transformer network with thin-plate spline transformation, which outputs the mapped coordinates of the control points without constraint. The architecture for the localization network that predicts the mapped

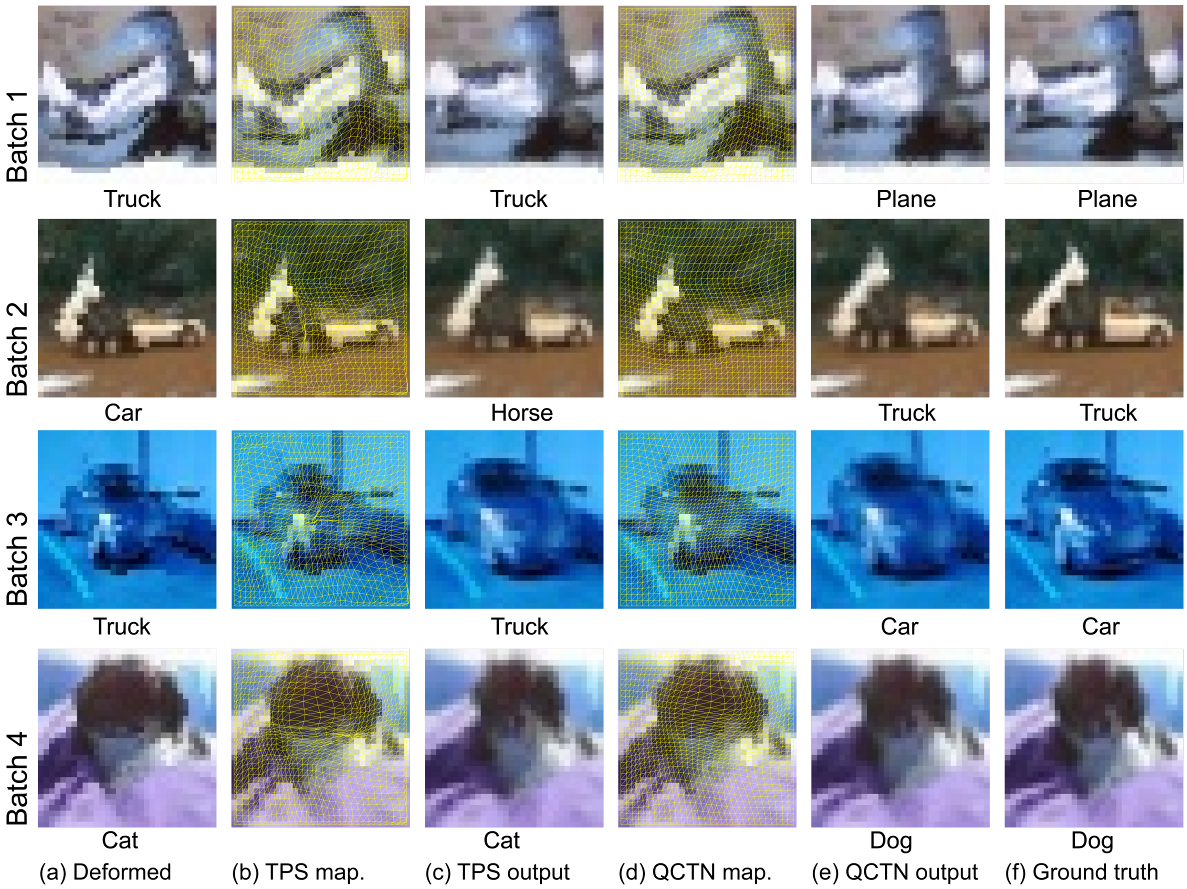

Figure 14: Classification result on CIFAR-10 with small non-rigid deformation. (a) the deformed images. (b) mapping generated by TPS-STN visualized on deformed image. (c) image recovered by TPS-STN. (d) mapping generated by QCTN visualized on deformed image. (e) image recovered by QCTN. (f) ground truth. The class names on the bottom of images in column (a)(c)(e) indicates the predicted class by baseline CNN, TPS-STN and QCTN respectively.

coordinates are the same as our coefficient estimator for a fair comparison. Though the motivation of thin-plate spline (TPS-STN) and quasi-conformal transformer shares the same motivation, the mapping generated by TPS-STN may not be a diffeomorphism and contains self-foldings.

The parameters in the experiment are set as we discussed at the beginning of this section, which is with a learning rate 0.00005 and clips the gradients value for the transformer layer with 10. It's worthy to mention that, gradients clipping is necessary for TSP-STN but not for our QCTN. Through we testing. The performance is similar for QCTN with or without clipping. But for TSP-STN, training without clipping the gradients would result in parameter explosion. We take the deep layer aggregation model(DLA,[32]) as the base classifier in this

Table 2: The results for classifying deformed CIFAR-10 with different scales by baseline CNN, TPS-STN and QCTN.

Method	Large Def.		Small Def.	
	Train	Test	Train	Test
CNN	94.55	75.84	99.13	83.06
STN-CNN	96.05	75.67	99.04	83.76
TPS-CNN	95.54	77.37	99.67	84.02
QCT-CNN	96.11	81.41	99.75	85.87

experiment.

Here we synthesize the deformation in two scales. Figure. 14 presents the small deformation, where some minor disturbance occurred. The large deformation is in Figure. 13, where the semantic meaning of the image is hard to distinguish. As illustrated in Table.2, the base CNN gets the lowest accuracy rate since the distortions bring some noise and degrade the images with information loss. For the method with transformer layers, our quasi-conformal transformer network obtain a better result than TPS-STN associated with mappings free of self-folding. From the visualized figures of the test dataset, images recovered by QCTN are more accurate and close to the original images, which helps human beings and neural network model to recognize the classes each image belong to.

5.3. Localization and Restoration on FashionMNIST. The localization f_l and restoration f_r can be composited into a single mapping $f_c = f_r \circ f_l$ as it's still a diffeomorphism. Thus, should be able to be learned in our Quasi-conformal transformer framework. In this section, with the deformation consisting of both affine transformation and elastic deformation assigned on the dataset FashionMNIST, we present the advantages of our QC-transformer network that can solve the localization and the restoration simultaneously in a single QC transformer layer. We also do a comparison with the spatial transformer network and its thin-plate spline variant as well as the pure classifier without any quasi-conformal or spatial transformer layers.

Table 3: The results for classifying deformed Fashion MNIST with different scales by baseline CNN, STN, TPS-STN and QCTN

Method	Large Def.		Small Def.	
	Train	Test	Train	Test
CNN	71.51	68.91	73.77	71.10
STN-CNN	80.86	76.11	82.57	78.83
TPS-CNN	84.29	80.82	86.13	83.59
QCT-CNN	84.18	83.84	86.48	85.79

The parameters set for this experiment are just like the previous, gradients clipping by 10

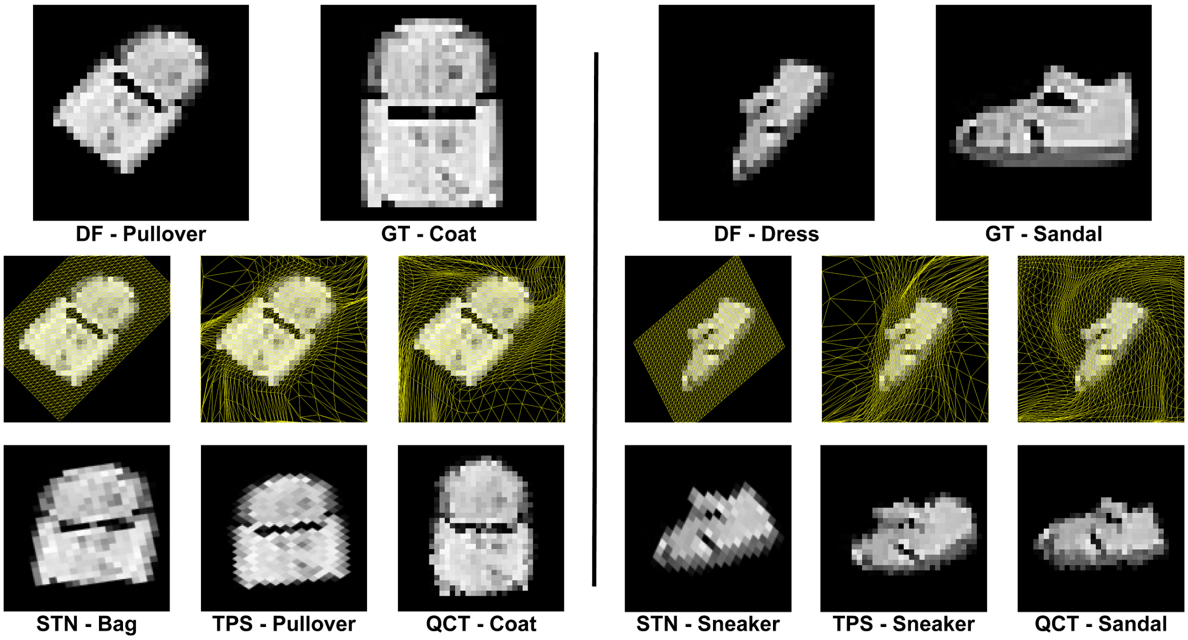

Figure 15: Classification result on Fashion MNIST with small non-rigid&affine deformation. Every batch follow the same illustration rule. As a example, DF-Bag present the deformed image and is predicted as bag by baseline CNN. GT-shirt indicates the ground truth image with the label as shirt. STN-Pullover, TPS-Sandal, QCT-shirt present the image recovered by STN, TPS-STN, QCTN and the predicted results respectively. The mapping on deformed image are visualized above them accordingly.

and the learning rate set to be 0.00005. Our methods outperform the other methods including the basic STN and its TPS variant as well as the baseline classifier which is the same as that in Section. 5.1.

As illustrated in Table. 3, our methods did the best on both deformation scales and achieve around 3% and 2% higher accuracy than that of TPS-STN. Illustrated in Figure.16 for large deformation and Figure.15 for small deformation, QCTN is able to recover the blurred and distorted image the best while simultaneously localizing the main objects in the image.

6. Conclusion and Future Work.

In this paper, we introduced the Quasi-conformal transformer network, which can be inserted into any part of a network. QCTN is capable to localize the regions that are important for the tasks. Besides, for images that contain distortions that may destroy the semantic meaning, QCTN can be trained to restore and recover the features for accomplishing tasks. More than that, since QCTN is a self-contained module, it can be inserted into any place of a neural network model to form up a new end-to-end trainable network model. Through the learnable transformation by QCTN, a deformable convolution can be performed by convoluting with regular rectangle on deformed feature map which is mathematically equivalent. Through the experiments, QCTN is proven to be able to assign not only spatial-invariant but also elastic deformation invariant to a network model. Besides,

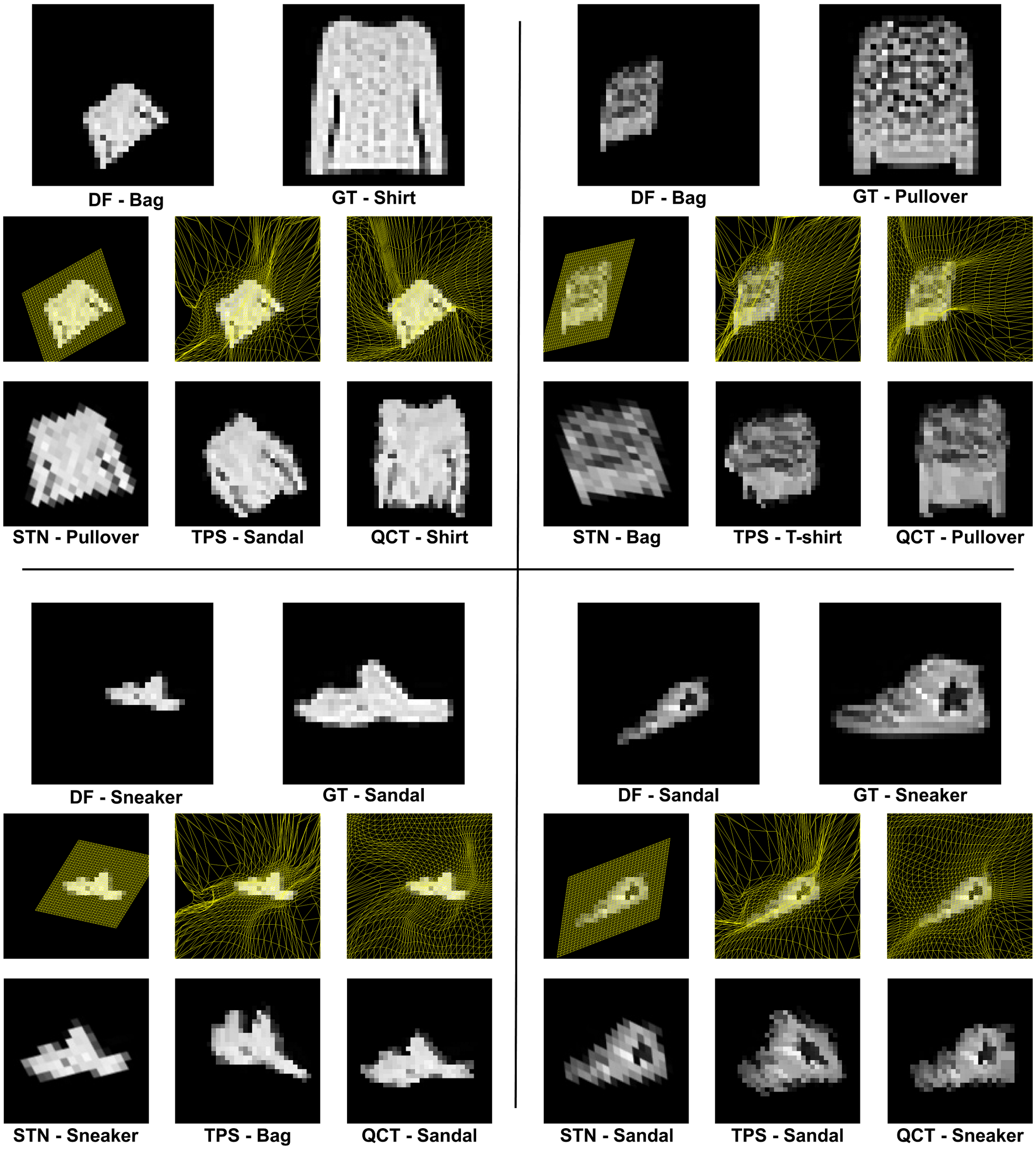

Figure 16: Classification result on Fashion MNIST with large non-rigid&affine deformation. Every batch follow the same illustration rule as Figure.15

compared to the similar work spatial transformer network and its thin-plate spline variant, our model is a point-wisely learnable transformation whose topology is guaranteed to be preserved by the proper constraight Beltrami coefficients. In our following works, we plan to extend the application of the Quasi-conformal transformer network. For example, on registration and segmentation. Besides, it would be interesting to assign bijectivity directly on the deformable convolution network[6] to see if preserving the topology of the deformed filter can help with the performance. Also, designing a bijective transformer for 3D volume deep learning would also be valuable, especially for medical images.

REFERENCES

[1] Q. Chen, Z. Li, and L. M. Lui, *A learning framework for diffeomorphic image registration based on quasi-conformal geometry*, CoRR, abs/2110.10580 (2021), https://arxiv.org/abs/2110.10580, https://arxiv.org/abs/2110.10580.
[2] G. Choi, Y. Liu, and L. M. Lui, *Free-boundary conformal parameterization of point clouds*, arXiv preprint arXiv:2010.15399, (2020).
[3] G. P.-T. Choi, K. T. Ho, and L. M. Lui, *Spherical conformal parameterization of genus-0 point clouds for meshing*, SIAM Journal on Imaging Sciences, 9 (2016), pp. 1582–1618.
[4] P. T. Choi and L. M. Lui, *Fast disk conformal parameterization of simply-connected open surfaces*, Journal of Scientific Computing, 65 (2015), pp. 1065–1090.
[5] K. Crane, U. Pinkall, and P. Schröder, *Robust fairing via conformal curvature flow*, ACM Trans. Graph., 32 (2013).
[6] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, *Deformable convolutional networks*, in iccv, 2017, pp. 764–773.
[7] J. D. Foley, A. Van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips, *Introduction to computer graphics*, vol. 55, Addison-Wesley Reading, 1994.
[8] X. Gu, Y. Wang, T. F. Chan, P. M. Thompson, and S.-T. Yau, *Genus zero surface conformal mapping and its application to brain surface mapping*, IEEE Transactions on Medical Imaging, 23 (2004), pp. 949–958.
[9] X. Gu and S.-T. Yau, *Global conformal surface parameterization*, in Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Eurographics Association, 2003, pp. 127–137.
[10] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in cvpr, 2016, pp. 770–778.
[11] M. Jaderberg, K. Simonyan, A. Zisserman, et al., *Spatial transformer networks*, Advances in neural information processing systems, 28 (2015), pp. 2017–2025.
[12] Y. Jeon and J. Kim, *Active convolution: Learning the shape of convolution for image classification*, in cvpr, 2017, pp. 4201–4209.
[13] N. Kanopoulos, N. Vasanthavada, and R. L. Baker, *Design of an image edge detection filter using the sobel operator*, IEEE Journal of solid-state circuits, 23 (1988), pp. 358–367.
[14] A. Krizhevsky, *Learning multiple layers of features from tiny images*, tech. report, 2009.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 25 (2012), pp. 1097–1105.
[16] K. C. Lam and L. M. Lui, *Landmark-and intensity-based registration with large deformations via quasi-conformal maps*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 2364–2392.
[17] Y. LeCun, Y. Bengio, et al., *Convolutional networks for images, speech, and time series*, The handbook of brain theory and neural networks, 3361 (1995), p. 1995.
[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324.
[19] Y. LeCun, C. Cortes, and C. Burges, *Mnist handwritten digit database*, ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist, 2 (2010).
[20] B. Lévy, S. Petitjean, N. Ray, and J. Maillot, *Least squares conformal maps for automatic texture atlas generation*, ACM transactions on graphics (TOG), 21 (2002), pp. 362–371.
[21] L. M. Lui, K. C. Lam, T. W. Wong, and X. Gu, *Texture map and video compression using beltrami*

representation, SIAM Journal on Imaging Sciences, 6 (2013), pp. 1880–1902.

[22] L. M. Lui, K. C. Lam, S.-T. Yau, and X. Gu, *Teichmuller mapping (t-map) and its applications to landmark matching registration*, SIAM Journal on Imaging Sciences, 7 (2014), pp. 391–426.

[23] W. Luo, Y. Li, R. Urtasun, and R. Zemel, *Understanding the effective receptive field in deep convolutional neural networks*, in Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, pp. 4905–4913.

[24] P. C. Ng and S. Henikoff, *Sift: Predicting amino acid changes that affect protein function*, Nucleic acids research, 31 (2003), pp. 3812–3814.

[25] J. Panetta, M. Konaković-Luković, F. Isvoranu, E. Bouleau, and M. Pauly, *X-shells: A new class of deployable beam structures*, ACM Transactions on Graphics (TOG), 38 (2019), pp. 1–15.

[26] J. M. Prewitt, *Object enhancement and extraction*, Picture processing and Psychopictorics, 10 (1970), pp. 15–19.

[27] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556, (2014).

[28] C. Y. Siu, H. L. Chan, and R. L. Ming Lui, *Image segmentation with partial convexity shape prior using discrete conformality structures*, SIAM Journal on Imaging Sciences, 13 (2020), pp. 2105–2139.

[29] Y. Soliman, D. Slepčev, and K. Crane, *Optimal cone singularities for conformal flattening*, tog, 37 (2018).

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, *Going deeper with convolutions*, in cvpr, 2015, pp. 1–9.

[31] H. Xiao, K. Rasul, and R. Vollgraf, *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*, CoRR, abs/1708.07747 (2017), http://arxiv.org/abs/1708.07747, https://arxiv.org/abs/1708.07747.

[32] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, *Deep layer aggregation*, in cvpr, 2018, pp. 2403–2412.

[33] D. Zhang and L. M. Lui, *Topology-preserving 3d image segmentation based on hyperelastic regularization*, Journal of Scientific Computing, 87 (2021), pp. 1–33.

[34] X. Zhu, H. Hu, S. Lin, and J. Dai, *Deformable convnets v2: More deformable, better results*, in cvpr, 2019, pp. 9308–9316.