SAPIENZA
UNIVERSITÀ DI ROMA

# Hypercomplex Generative Deep Learning for Image Modality Translation


Facoltà di Ingegneria dell'Informazione, Informatica e Statistica

Corso di Laurea Magistrale in Engineering in Computer Science

Candidate

Luigi Sigillo
ID number 1761017

Thesis Advisor

Prof. Danilo Comminiello

Co-Advisor

Eleonora Grassucci



Academic Year 2020/2021

Thesis defended on 17 January 2022
in front of a Board of Examiners composed by:

Prof. Daniele Nardi (chairman)
Prof. Danilo Comminiello
Prof. Giuseppe Antonio Di Luna
Prof. Riccardo Lazzeretti
Prof. Andrea Marrella
Prof. Simone Scardapane

---

**Hypercomplex Generative Deep Learning for Image Modality Translation**
Master's thesis. Sapienza – University of Rome



This thesis has been typeset by LaTeX and the Sapthesis class.

Version: January 11, 2022

Author's email: luigisigillo30@gmail.com

*A nonno Fabio*

# Acknowledgments

*Ci terrei a ringraziare il Prof. Danilo Comminiello che da subito mi ha messo a mio agio e dato grande fiducia e disponibilità nel compiere un progetto così interessante e ambizioso, essendo tale non sono infatti mancati dei problemini e insieme alla dottoranda Eleonora Grassucci si sono sempre dimostrati disponibili nel cercare di venirne a capo insieme a me.*

*Vorrei anche ringraziare la mia famiglia che mi ha sempre sopportato in questi anni, che si sono rivelati alquanto impegnativi, visto anche l'avvento della pandemia, la loro presenza li ha resi più felici.*

*Nonostante questa situazione ci abbia costretto a passare poco tempo insieme dal vivo, ho condiviso bei momenti virtuali durante le lezioni e le sessioni di studio con il gruppo dei "Sapienti", dove ho anche conosciuto dei veri fan di Eraclito!*

*Questi anni di studi e di vita si sono rivelati, in alcuni momenti, difficili da digerire e superare, ma per mia fortuna, ho sempre avuto accanto Qualcuno*  *che, durante i nostri mille momenti passati insieme, mi ha mostrato come la logica booleana nella vita può essere sopraffatta da quella fuzzy.*

*Ringrazio, anche questa volta, i "Nottambuli+1" che da anni mi accompagnano in ogni viaggio e lo hanno fatto perfino durante questi tempi bui, illuminandoli.*

*Non posso non menzionare anche i miei compagni di giochi durante le serate pandemiche, seppur a distanza, mi hanno sempre fatto sembrare questa situazione come normale.*

*Non ho fatto nomi e cognomi, ma ognuno a modo suo, in questi due anni, ha fatto parte di questo viaggio, che però sarà solo l'inizio.*

*Questi anni mi hanno fatto capire che non dobbiamo mai dare niente per scontato, perchè, da un momento all'altro, una persona, dall'altro lato del globo, potrebbe mangiarsi un pipistrello. Quindi non rimaniamo nella nostra comfort zone, sperimentiamo sempre cose nuove e buttiamoci nelle occasioni, non si sa quante ne avremo.*

*Per riassumere in breve, prendo in prestito le parole di Virgilio e Orazio:*

***tempus fugit, carpe diem**.*

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Image-to-image translation task is the translation of one possible representation of a scene into another, predicting pixels from pixels. Image-to-image translation problem can be solved with the use of conditional generative adversarial networks, they learn the mapping from input image to output image and learn a loss function to train this mapping. The novelty lays in the fact that we can apply the same generic approach to different problems with the same loss formulations. With this approach it is possible to synthesize photos from label maps, reconstructing objects from edge maps, and colorizing images as stated in [19].

This problem is extended to the task of Image Modality Translation, with the aim of translating images from a source modality to a target one. If we think about real world images it is common that they have multiple modalities i.e RGB image associated with the same image but infrared. The key factor is that those images, even if are in different domains, shares the same information for what concern edges textures and others structures. This is the reason why modality translation has recently aroused, also with the help of the generative deep learning, in different tasks such as the image super resolution [8] or denoising [23]. IMT could be really useful in medical imaging, in fact there are diverse imaging protocols varying in characteristics and applications, like magnetic resonance imaging (MRI), that can be divided in different modalities: T1-weighted, T2-weighted. Multi-modality MR images are really useful to diagnostic human diseases, naturally having complementary information from multi-modalities helps doctors to do a correct diagnosis. In a real world

scenario unfortunately those modalities are missing or inconsistent causing in some cases errors in diagnosis and treatment, from this problem arises the medical image synthesis. It is considered as the mapping between a target-modality image and its source modality. Conditional GAN models have been applied to 2D image synthesis, we have works in retinal images [45] or CT images [17] and MR images [5].

## 1.1 Contribution

IMT tasks require sophisticated models, composed by multiple networks, thus are generally really huge in terms of space and parameters to train, so to obtain realistic results there is the need of large computational resources. IMT are, as said before, in the family of the generative models, in general those models understands the inner structure of data without relying on label classes, they works well with data like images using convolutional layers. Convolution highlights relations between different channels in the images, and recent works [12] demonstrates how to exploit this relations in an hypercomplex domain using quaternions. Hypercomplex methods allow exploiting the correlations of multidimensional signals, such as color images, to reduce the complexity of neural architectures. In this work we will use two hypercomplex approaches:

- A classical quaternion approach.
- A parameterized hypercomplex approach.

Quaternions are composed of one real part and three imaginary part, this property is exploited for multi-layered structured data, when using quaternions, Hamilton multiplication between quaternions is used to substitute convolutions. Using the canonical quaternion approach saves parameter, having only 1/4 of learnable parameters, maintaining in most of the cases the performance in various applications. The reason why a parameterized hypercomplex approach is exploited is the fact that hypercomplex space, like the quaternion domain, generally exists at predefined dimensions (4D, 8D, 16D). The aim so is to have more flexible models using hypercomplex multiplications parameterizing them meaning that models learns multiplications rules from data even if those rules are not predefined as presented in [42].

Hypercomplex deep networks have many applications since quaternion-like data are commonly used, for example images and so when dealing with the Image Modality Translation problem. The goal of this master thesis is to see the potential of hypercomplex networks applied to conditional GANs through experiments, comparing known models for Image to image translation and Image modality translation to their hypercomplex extensions. In order to accomplish this objective we introduced for the first time in literature the quaternion instance normalization and the paremeterized hypercomplex transposed convolution. This a first study on quaternionic IMT models, more specifically on the hypercomplex StarGAN and TarGAN models, moreover we present further studies on the weight initializations for those hypercomplex layers and how this can boost the performances of hypercomplex models.

## 1.2 Thesis overview

In particular the thesis is divided in six chapters:

- **Chapter 2 Background:** we will present the problem of image modality translation and briefly what are GANs
- **Chapter 3 Hypercomplex Deep Learning:** we will see the fundamentals of quaternion algebra and how are applied to deep convolutional neural networks.
- **Chapter 4 Proposed Approaches:** an approach to build an hypercomplex model of StarGAN v2 [4] will be presented, introducing for the first time a quaternion instance normalization, a normalization commonly used when dealing with style transfer. We will see how weights initialization is a crucial part whenever doing deep hypercomplex learning and how this affects the parameterized hypercomplex approach. Then we will present, for what concern the IMT task, an hypercomplex version of the TarGAN [2], here we will introduce a particular approach to the problem of mono channel biomedical images, and their issue whenever we are in a hypercomplex space.
- **Chapter 5 Experimental Results:** Finally the results of the presented approaches are presented, showing both the qualitative and the quantitative

analysis. For both of them the hypercomplex approaches are really interesting reaching sometimes SOTA performances, and saving an huge number of parameters.

- **Chapter 6 Conclusions:** The conclusion will sum up the work done and my future ideas to explore more this continuously growing area of deep learning.

# Chapter 2

# Background

This chapter introduces some background material and the state-of-the-art literature for our problem, we will explain the main concepts behind Generative Deep Learning.

## 2.1 Generative Deep Learning

A generative model is used to solve a density estimation task, it learns unsupervised a data distribution: it estimates the probability density function $p(x)$ of a random variable $X$, given a collection of observations $\{x1, x2, ...\}$. We want to find a model that satisfy this condition: $p_{model}(x, \theta) \sim p_{data}(x)$ Using the maximum likelihood function:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{i}^{N} \log p_{model}(x_i, \theta) \tag{2.1}$$

This finds $p^*_{model} = p_{model}(\theta^*, x)$, it generates samples similar to the training set ones. A neural network can generalize the definition of the $p*_{model}$ generating data from that distribution and using *adversarial learning* it is possible to train another neural network to discriminate the output of $p^*_{model}$ from $p_{data}(x)$.

## 2.2 Generative Adversarial Networks

Ian Goodfellow in 2014 firstly introduced the concept of generative adversarial networks [11]. The problem of learning patterns of data and the consequently generation of new samples similar to the training set, could be in some sense simplified

if we split the problem in two different models: a generator and a discriminator. As we could imagine, the generator learns to generate new samples while the discriminator is a classifier that learns to distinguish real samples from the generated ones. The novelty is that the training of these models is done together in a zero-sum game, with this value function:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))] \quad (2.2)$$

Where

- $z$ is the input noise coming from $p_g(z)$
- $G(z)$ is a differentiable function representing fake data
- $D(x)$ is a differentiable function representing the probability that $x$ is a real sample

This means that the generator became more precise with the help of the discriminator, that on the other hand will start to fail in recognizing the real samples from the fake ones.
The generator takes as input some random noise, a vector of fixed length, randomly from a Gaussian distribution, and try to generate samples in the desired domain. The vector is taken from what is called the latent space, so a projection of a data distribution, meaning that those are important random variables in our model, but are not directly observable. So the generator finds meaningful information in the latent space and use that information to generate new and also different samples from the training set.

As said in [10] one could think that the generator is like a counterfeiter that tries to produce fake money and the discriminator is the police officer that tries to find out if the money given by the counterfeiter is real or fake, see fig 2.2. This is the part where we see the concept of *adversarial* in the GAN name, because of this zero-sum game between the two adversaries: the generator and the discriminator. The discriminator is not rewarded if it correctly guess weather a input is real or fake, and the generator gets penalized when fails to fool the discriminator. While if the

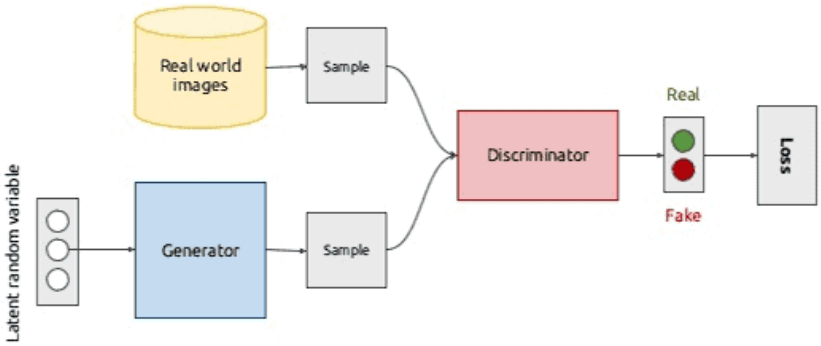


**Figure 2.1.** Architecture of a GAN

generator succeed in fooling the discriminator gets rewarded and the discriminator penalized. When we speak about reward and penalties, those means an update to the model parameters.

### 2.2.1 Training of a GAN

**Algorithm 1** Update rule of a GAN

**for** n training iterations **do**
  **for** k steps **do** ▷ Update Discriminator
    Sample $\left\{x^{(1)} \dots x^{(m)}\right\}$ from $p_{\text{data}}\left(x\right)$
    Sample $\left\{z^{(1)} \dots z^{(m)}\right\}$ from $p_z(z)$

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\mathbf{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)\right]$$

  **end for**
  **for** 1 step **do** ▷ Update Generator

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\mathbf{z}^{(i)}\right)\right)\right)$$

  **end for**
**end for**

The parameters can be updated using SGD (Stochastic Gradient Descent) or other algorithms. The step for updating the discriminator are usually more than the 1 mentioned for the generator. This as the original work [11] said, could lead to a better performance of the network because of the discriminator staying more close

to the optimum $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x)+p_g(x)}$ and forcing the generator to produce better and better results. There can be a state where the two models cannot improve because the generator is generating realistic data and the discriminator is unable to see the difference between real and fake samples, this is called nash equilibrium and represent the optimal solution $p_g = p_{data}$. In this case the two players (Generator and Discriminator) reached a point where they choose the best strategy for their selves, and not change their strategy basing on the opponent moves. This state will not bring convergence to the models, and this is an issue of the GAN.

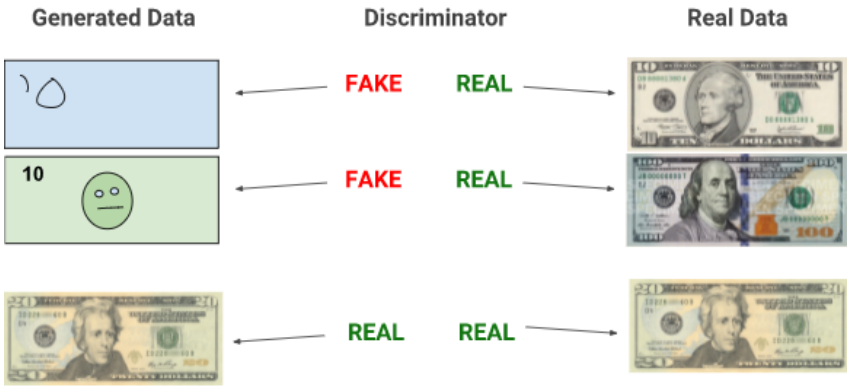


**Figure 2.2.** Training of a GAN

### 2.2.2 Conditional GANs

The original idea of the GAN is the one presented in figure 2.1, but clearly in the literature there are at this point a number of extension and updates that boost the performances of the original idea. One of those is the idea of conditional GAN, so we are conditioning the random vector that is provided to the generator from the latent space. The condition could be represented by labels [26] and given to both of the models, or more importantly it could also be an image [19], from the training domain. These bring to the category of the GAN to perform the task of image-to-image translation, the ability to translate images across domains, such as male to female, day to night and so on and so forth. GANs for image-to-image mappings have been applied before [31] for inpainting, but not using conditional GAN in the sense that they use, for example L2 regression, to force the output to

be conditioned. So the value function of the GAN 2.2 becames 2.3, so the prior input noise $p_z(z)$ and y are combined in a joint hidden representation, for both the generator and the discriminator.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))] \quad (2.3)$$

In [19] they used a U-Net for the generator and a PatchGAN classifier for the discriminator, the particularity is that this penalizes structure considering image patches, this is shown in 2.3.

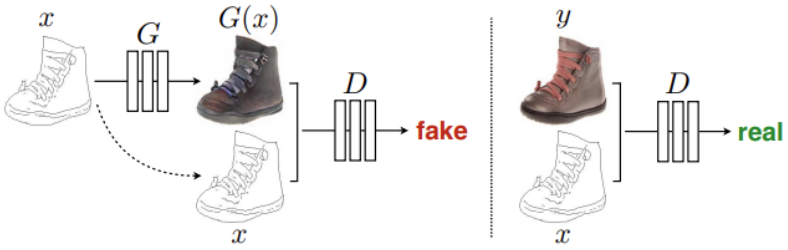


**Figure 2.3.** Training of a conditional GAN to map edges to photo [19]

## 2.3 Image Translation Tasks

### 2.3.1 Image-to-Image Translation

When speaking about image-to-image translation, we want a model that learns mapping between different visual domains and that satisfies the following properties:

1. Diversity of generated images

2. Scalability over multiple domains

Existing models partially solve those issues, but not completely and not together. We will see in chapter 4 a model that can handle both issues and reaches state of the art performances. For domain, when image-to-image translation is involved, we mean images that are grouped because have unique characteristic and appearance, called *style*. The gender of a person could be a valid domain and in that case styles can be represented by beard, hairstyle and similar. A typical solution to address the

style diversity problem is to inject a low-dimensional latent code into the generator, in this way the decoders, inside the generator, reached to interpret correctly this code as a style code. The problem of those models is that they can only learn mapping between two domains and not reaching the scalability, in this way we have to train $K(K-1)$ generators, where $K$ is the number of domains.

The previous version of Stargan [3] is one of the models that firstly tries to address those issue, but on the other hand now the issue of a deterministic mapping raise, in fact this model not faces the problem of the multi-modality of the data distribution. This problem is embedded in the solution proposed, in fact the fixed label, representing the domain, received by the generator force the generator to produce the same output for a domain given an input image.

Unpaired image to image translation as presented in [48] in a model called **cycle-GAN** can be considered a pioneer work since this method is used as backbone for a large amount of networks that do image to image translation. The novelty introduced in this work is that the mappings of an IIT model should be reversible, so we can revert the function $\mathbf{G} : \mathbf{X} \rightarrow \mathbf{Y}$. This can be achieved using a cycle consistency loss that forces $F(G(x)) \approx G(Y(y)) \approx y$. For the images from domain Y and in a similar loss also for domain X with $D_x$, this holds:

$$\mathcal{L}_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{dau}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{dau}(x)}[\log D_Y(y)] + \mathbb{E}_{x \times p_{dau}(x)}(1 - D_Y(G(x)))]$$

The cycle consistency loss enforce forward and backward consistency in this way:

$$\mathcal{L}_{\text{cyc}}\,(G, F) = \mathbb{E}_{x \sim p_{\text{data}}\,(x)}\,[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}\,(y)}\,[\|G(F(y)) - y\|_1]$$

From figure 2.4 it is visible the architecture of this model, the intuition is that if

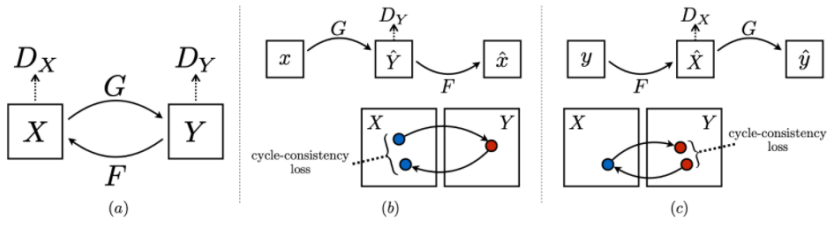


**Figure 2.4.** CycleGAN [48]

we start from a domain $X$ and then translate to a domain $Y$ we should arrive to $X$ again starting from $Y$.

### 2.3.2 Image Modality Translation

Image modality translation is a specialization of image to image translation. It is in fact really similar to the previous one and can be used to obtain images of similar modalities. A generic model that does IMT learns to generate translated modalities from given modalities in MR images, leveraging the conditional generative adversarial networks (cGANs) that we have discussed earlier. The need of this specialization comes from the fact that image to image translation provides a generic solution for problems in natural scenes i.e. mapping images to edges, of course the arise of conditional GANs, accelerated this interest and this process.
Image modality translation is a specialization of image to image translation. It is in fact really similar to the previous one and can be used to obtain images of similar modalities. The problem of IIT is that there are diverse output given a single condition, this could be avoided when the we focus on both the conditioning vector and the input noise as done in [24], where they propose a model that has a generator that explore minor modes during training. In recent works [1] IMT is used to infer the 3D shape of an object starting from a 2D image, so there is the tranlsation between those two different modalities, the high level representation is showed in 2.5.

This is applicable to many biomedical cases, so working with MRI or CT images.

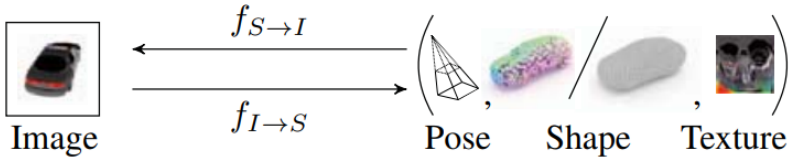


**Figure 2.5.** Cycle-Consistent Generative Rendering for 2D-3D Modality Translation [1]

A generic model that does IMT learns to generate translated modalities from given modalities in MR images, leveraging the conditional generative adversarial networks (cGANs) that we have discussed earlier. The need of this specialization comes from the fact that image to image translation provides a generic solution for problems

in natural scenes i.e. mapping images to edges, of course the arise of conditional GANs, accelerated this interest and this process. The main problem still is that the literature focused on natural scenes, but in medical scenario there are different needs, so the idea to introduce image modality translation to do MRI/CT cross modality generation, as done by [40].

An approach for IMT to do MRI synthesis based on conditional generative adver-

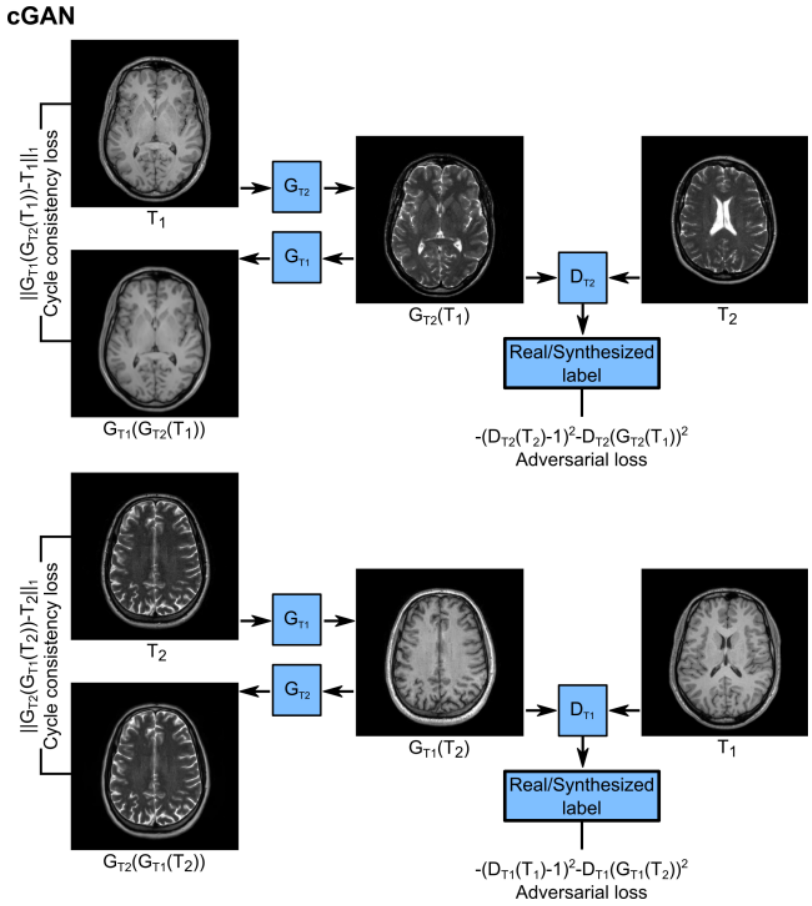


**Figure 2.6.** cGAN. Generators tries to learn how to produce $T_2$ or $T_1$ weighted images and make them indistinguishable from the discriminator [40]

sarial networks is presented here [5]. In this work the model is trained to preserve intermediate to high frequency details using the canonical adversarial loss and uses perceptual losses for registered multi contrast images and a cycle consistency loss for unregistered images.

A significant improvement was done in [16] where the CycleGAN approach is ex-

tended adding a gradient consistency loss to improve the quality of the images generated at their boundaries. This is a problem since now in fact this approach was applied to unpaired CT and MR images of the head, but not in a pelvic region where there are more problems since it is a part where there are joints and muscles. There are in literature models [27] that does IMT to generates nonexistent MRA from existing MR multi-contrast, clearly this task is adding more information on a normal MR that does not take into account vascular anatomy and related diseases. The vessels are very important in this translation so they introduced a loss term dedicated to reproduce properly the vascularities. This is done incorporating steerable filter responses of the generated and reference images inside a Huber function loss term, they proposed the steerable GAN (sGAN) showed in 2.7.

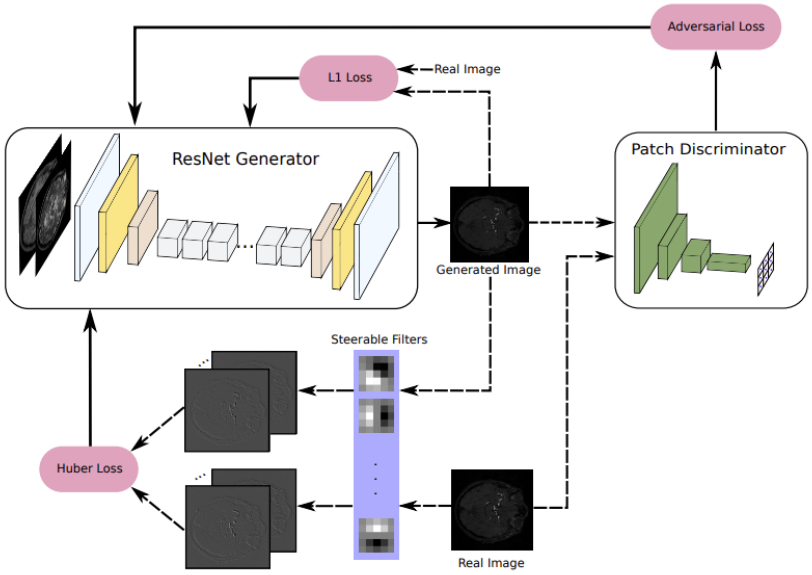


**Figure 2.7.** sGAN architecture [27]

The quality of the generated image is measured with three loss functions, the classical adversarial loss of the **PatchGAN** discriminator, the reconstruction loss to measure similarity between original images and generated ones and the steerable filter through a Huber loss function.

In [41], there is the propose instead to enforce the IMT model to pay attention to the edges, called edge-aware generative adversarial networks (Ea-GANs) for cross-modality MR image synthesis. A generator-induced Ea-GAN (gEa-GAN)

and a discriminator-induced Ea-GAN (dEa-GAN) are proposed as solutions: the first includes the edge information via its generator, while the latter from both the generator and the discriminator in this way the edge similarity is adversarially learned.

We can conclude that in general medical image synthesis it's of great benefit because we can have an image modality scan without doing the actual scan. In chapter 4 the **TarGAN** [2], a state of the art model that does both segmentation ad translation, will be presented and more details on how this type of frameworks works will be discussed.

# Chapter 3

# Hypercomplex Deep Learning

## 3.1 Quaternions

Deep learning models that deal with images, make a large use of convolutional operations. Convolution can capture spatial dependencies like the shapes of an image. Taking into account that images are multidimensional data, represented by pixels on three channels (Red, Green and Blue), we want to capture the intra-channel relations, meaning for example a color generated by the mixing of the primary three. To capture these pixel relations in the color space we are using real valued approaches, but in those approaches we consider the three entities as one input, loosing the information that they are independent.

We are using different kind of information, local and global dependencies, at the same level, so the model has to learn the difference during training. There are examples in the literature of this phenomenon, in [29] the network failed to capture the color if it is trained on greyscale images, so it is not able to generalize. To deal with that we need a different approach, with respect to the real valued one: the first idea has been the neural network with multi-dimensional numbers, for instance complex numbers. They have been proven to be really impressive with respect to their real counterpart whenever there are two dimensional inputs. The problem with our task is that those network are limited to a two-dimensional space while we are looking for a solution that fits well for an image task.

To this purpose we will explore the quaternion hypercomplex domain, in fact they

are made by a real part and three separate imaginary components.

### 3.1.1 Fundamentals of Quaternion Algebra

Quaternions have been discovered by William R. Hamilton, the initial idea of Hamilton starts from the need of representing points in a three dimensional space, so numbers with one real part and two complex parts. Then the idea of adding another complex part come to mind and that brings to the fundamental formula:

$$i^2 = j^2 = k^2 = ijk = -1$$

The quaternions were abandoned in favour of vector analysis because it describes the same concept but in an easier way. Nowadays with the high computational power, the quaternions starts to be used again in many fields, solving problem as the gimbal lock with rotations and orientations using rotation matrices and Euler angles.
Quaternion are hypercomplex numbers composed by a one real part and three imaginary parts, they lay in the four dimensional vector space $\mathbb{H}$. A quaternion number is represented as

$$q = r + x\boldsymbol{i} + y\boldsymbol{j} + z\boldsymbol{k} \tag{3.1}$$

where $r$, $x$, $y$, $z \in \mathbb{R}$ and **i,j,k** are the unit axis vector of orthonormal basis in $\mathbb{R}^{\not|\not|}$.
Another way of representing a quaternion is $q = (r, \boldsymbol{v}) = r + \boldsymbol{v}$.
Quaternions follows non-commutative multiplication rules:

$$\begin{aligned} ij &= k \\ jk &= i \\ ki &= j \\ ji &= -k \\ kj &= -i \\ ik &= -j \end{aligned} \tag{3.2}$$

Addition and subtraction are defined as

$$q = q_1 \pm q_2 = (r_1 \pm r_2) + (x_1 \pm x_2)i + (y_1 \pm y_2)j + (z_1 \pm z_2)k \tag{3.3}$$

The key operation supported by the quaternions is the **Hamilton product** denoted as $\bigotimes$. There are actually two form of writing this operation, the matrix multiplication is more readable in my opinion:

$$q = \begin{bmatrix} r_1 & -x_1 & -y_1 & -z_1 \\ x_1 & r_1 & -z_1 & y_1 \\ y_1 & z_1 & r_1 & -x_1 \\ z_1 & -y_1 & x_1 & r_1 \end{bmatrix} \begin{bmatrix} r_2 \\ x_2 \\ y_2 \\ z_2 \end{bmatrix} \tag{3.4}$$

Other common operation on a quaternion are its conjugate as shown in 3.5, its norm shown in 3.6 and its inverse 3.7.

$$\overline{q} = r - v = r - xi - yj - zk \tag{3.5}$$

$$|q|| = \sqrt{q\overline{q}} = \sqrt{r^2 + x^2 + y^2 + z^2} \tag{3.6}$$

$$q^{-1} = \frac{\overline{q}}{||q||^2} \tag{3.7}$$

### 3.1.2 Deep Quaternions Neural Networks

The quaternions neural networks succeed to learn well the local relations, missing in a real valued approach as stated in [28].

For example images are RGB matrices, so with 3 channels, can be viewed as a quaternion input like $q = 0 + R\boldsymbol{i} + G\boldsymbol{j} + B\boldsymbol{k}$. In this way we can apply all the quaternion rules and operations like the Hamilton product, using a reduced number of parameters. Some layer will be presented in the next section, to see the differences between the real valued and the quaternions.

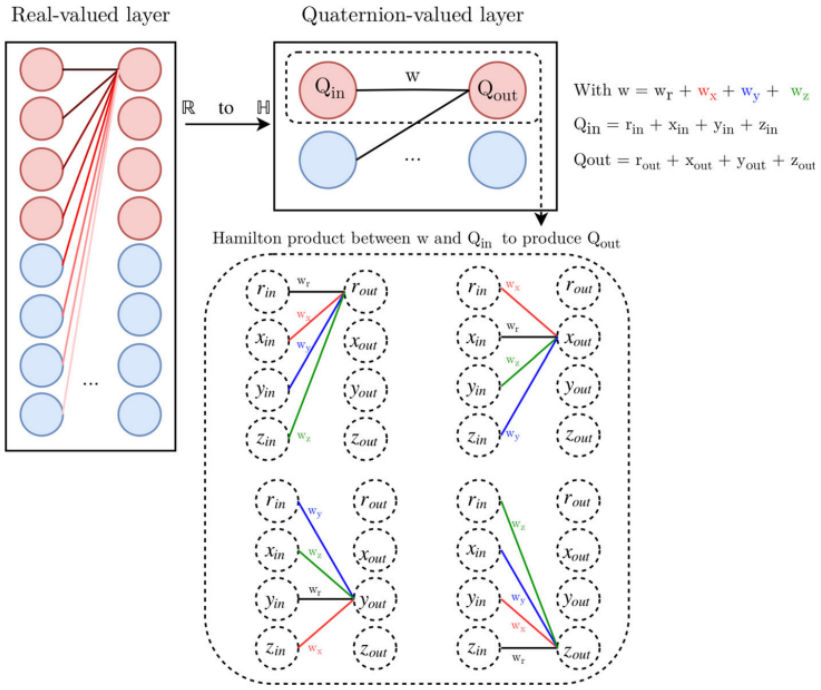


**Figure 3.1.** Quaternion weight sharing using Hamilton product compared to the real valued approach [28]

#### 3.1.2.1 Quaternion Fully Connected Layer

In a fully connected layer the input vectors are connected to all the neurons or the activation units. This can be seen as a matrix multiplication

$$f(y^{l-1}, W^l) = W^l y^{l-1} + b^l \tag{3.8}$$

where $y^{l-1}$ is the output of the previous layer and $b^l$ is the bias.
The weights are independent between each other so in the computation of the gradient the derivative is done with respect to all of the parameters. As said previously in the quaternion domain the multiplication is substituted by the Hamilton product, so the previous equation 3.8 becomes

$$f(y_q^{L-1}, W_q^l) = W_q^l \otimes y_q^{l-1} + b_q^l \tag{3.9}$$

The weights matrix in 3.9 is composed by $W_q^l = W_r^l + W_x^l i + W_y^l j + W_z^l k$.

$$W = \begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} & W_{1,4} \\ W_{2,1} & W_{2,2} & W_{2,3} & W_{2,4} \\ W_{3,1} & W_{3,2} & W_{3,3} & W_{3,4} \\ W_{4,1} & W_{4,2} & W_{4,3} & W_{4,4} \end{bmatrix} W_q = \begin{bmatrix} W_r & -W_x & -W_y & -W_z \\ W_x & W_r & -W_z & W_y \\ W_y & W_z & W_r & -W_x \\ W_z & -W_y & W_x & W_r \end{bmatrix} \tag{3.10}$$

The comparison of the two weights matrices is shown in 3.10.
So the matrix $W$, real valued, has this structure having 16 different parameters that are multiplied for the input, while $W_q$ has only 4 parameters.

#### 3.1.2.2 Quaternion Convolutional layer

Also in this operation there is the substitution of the multiplication with the Hamilton product,

$$y_{ij}^l \equiv \sum_m \sum_n w_{mn}^l \; y_{i+m,j+n}^{l-1} + b_{ij}^l \tag{3.11}$$

so the equation becomes:

$$y_{ij}^i = \sum_m \sum_n w_{mn}^l \otimes y_{i+m,j+n}^{l-1} + b_{ij}^l$$

To see the quaternion advantages better, the convolutional operation can be viewed in its matrix form:

$$\begin{bmatrix} Y_r \\ Y_x \\ Y_y \\ Y_z \end{bmatrix} = \begin{bmatrix} W_r & -W_x & -W_y & -W_z \\ W_x & W_r & -W_z & W_y \\ W_y & W_z & W_r & -W_x \\ W_z & -W_y & W_x & W_r \end{bmatrix} * \begin{bmatrix} X_r \\ X_x \\ X_y \\ X_z \end{bmatrix} \tag{3.12}$$

We are arranging the matrix $W$, as shown in 3.12, reusing the real valued components to build the real-valued matrix. Then to model the imaginary units we have to insert some plus or minus in the right position to perform the product. $W$ is composed by convolutional filters, we can organize those filters following the Hamilton product rule, moreover due to this algebra rule, each components of the input is multiplied for each submatrix, so the relations among the input components are captured by the

weight matrix since weight submatrix are reused and multiplied for each component.

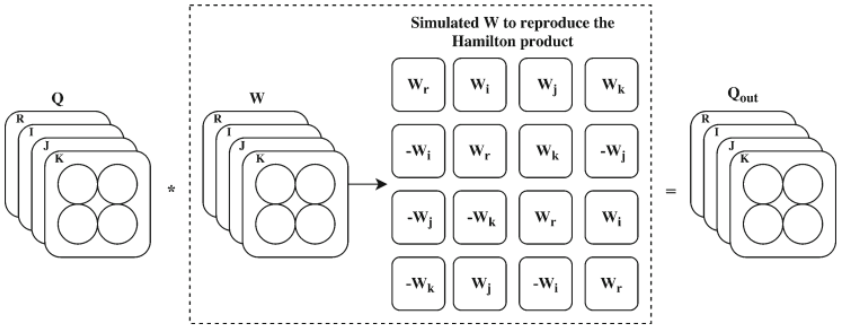


**Figure 3.2.** Convolution in the quaternion domain [30]

#### 3.1.2.3 Quaternion activation functions

Activation functions are required when we are dealing with non-linear systems, they are used on the output of other layers. One of the most used activation function, the ReLU, is applied component-wise resulting in an output that has the same shape of the input vector but with different values. In the quaternion domain, this functions are applied separately on each of the component of a quaternion. Firstly introduced here [34], they are called split activations functions.

$$f(q) = f(r) + f(x)\boldsymbol{i} + f(y)\boldsymbol{j} + f(z)\boldsymbol{k}$$

Where $f$ is a generic activation function, keep in mind that those activation functions are not quaternion based, so they not consider relation between components, so it is a compromise that we have to accept.

#### 3.1.2.4 Quaternion Batch Normalization

We present a way to handle normalizations in quaternion valued neural networks, even if we will not use in our experiments models that use this normalization. we will present later on, in the next chapter, how we exploited the instance normalization and its counterpart in the quaternion domain.

The approach is similar to the real one, the mean $\mu$ and the covariance $\sigma$ are

computed across the batch distribution of the data, and afterwards there is the scaling and translating through $\gamma$ and $\beta$. The difference its in how these values are calculated, in fact the mean is a quaternion number, so not a scalar: it is computed as:

$$\mu_q(q) = \frac{1}{N}\sum_{t=1}^{N}(qt) = \overline{r} + \overline{x}i + \overline{y}j + \overline{z}k$$

it is built by the means of every real part of the quaternion numbers.
The covariance is a real scalar computed with the mean of the sum of the squared components of the quaternionic mean:

$$\sigma_q(q) = \frac{1}{N}\sum_{t=1}^{N}(\Delta r^2 + \Delta x^2 + \Delta y^2 + \Delta z^2)$$

So the formula of the batch normalization can be sum up as:

$$\begin{aligned} QBN(q_i) &= \gamma\left(\frac{q_i - \mu_q(q)}{\sqrt{\sigma_q(q) + \epsilon}}\right) + \beta \\ &= \gamma\left(\frac{r_i - \overline{r}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}\right) + \gamma\left(\frac{x_i - \overline{x}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}\right) + \gamma\left(\frac{y_i - \overline{y}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}\right) + \gamma\left(\frac{z_i - \overline{z}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}\right) + \\ &+ (\beta_r + \beta_x + \beta_y + \beta_z) \end{aligned}$$

$\gamma$ is a trainable parameter that tells how much to scale and $\beta$ is a trainable quaternion parameter representing how much shift is needed.

## 3.2 Parameterization of Hypercomplex Multiplications

Following the success achieved by the quaternion applied in neural networks field, there is a recent novelty in this approaches in the hypercomplex space. The reason behind this novelty is because of the limit that hypercomplex space has predefined dimensions (4D, 8D and 16D), so this affect in a bad manner the usage of those models.
In this section we will present the paremeterizing of the hypercomplex multiplications [42] that learn multiplication rules even if not in a predefined set. This method has the ability to learn in a $n$D hypercomplex space, so brings the flexibility that is

missing inside the hypercomplex neural networks and thus uses only $1/n$ parameters. The Hamilton product permits to save $1/4$ of the learnable parameters with respect to the real valued approach, nonetheless it is substituted by the Kronecker's product in this new hypercomplex approach. The interest in this field raised because of the deep usage in neural networks of the fully connected layers, this layers are the fundamental to build more complex and sophisticated layers, for example in the work done to build the fantastic transformers architecture [36].

### 3.2.1 PHM Layers

The layers that will be presented in the next section will substitute the multiplication with the sum of Kronecker products.

#### 3.2.1.1 PHM Fully Connected Layer

The canonical fully connected layer takes as input a vector $x \in \mathbb{R}^d$ and returns a vector $y \in \mathbb{R}^k$, the complete equation is the following:

$$\mathbf{y} = \mathrm{FC}(\mathbf{x}) = \mathbf{W}\mathbf{x} + b \tag{3.13}$$

the weight matrix is $\mathbf{W} \in \mathbb{R}^{k \times d}$ and the bias is $\mathbf{b} \in \mathbb{R}^k$. The 3.13 becomes:

$$\mathbf{y} = \mathrm{PHM}(\mathbf{x}) = \mathbf{H}\mathbf{x} + b,$$

so the weight matrix is replaced by $\mathbf{H} \in \mathbb{R}^{k \times d}$ and as anticipated it is composed by a sum of Kronecker products.
The Kronecker product is an operation between two matrices that results in a block matrix, it can be considered a generalization of the outer product from vectors to matrices, it is defined as:

$$\mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{11}\mathbf{Y} & \dots & x_{1n}\mathbf{Y} \\ \vdots & \ddots & \vdots \\ x_{m1}\mathbf{Y} & \dots & x_{mn}\mathbf{Y} \end{bmatrix} \in \mathbb{R}^{mp \times nq},$$

$x_{i,j}$ is the element of $X$ at its $i^{th}$ row and $j^{th}$ column. The key point in why there is the $\mathbf{H} \in \mathbb{R}^{k \times d}$ matrix and how it is composed lays in this concept: let's $k$ and $d$ be divisible by a $n \in \mathbb{Z}_0$ arbitrarily chosen, for each $i = 1, 2, .., n$ there are parameters matrices $\mathbf{A}_i \in \mathbb{R}^{n \times n}$ and $\mathbf{S}_i \in \mathbb{R}^{\frac{k}{n} \times \frac{d}{n}}$ and the sum of the Kronecker's product between these two matrices form $H$ as shown in 3.14.

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{A}_i \otimes \mathbf{S}_i. \tag{3.14}$$

To see why we have the reduction in the number of parameters by a factor of $n$, we notice that in the original fully connected layer 3.13 $W$ dominates the parameterization and the number of parameters is $O(kd)$, while in the PHM version the dominant matrix is $\mathbf{H}$ where the numer of parameters is in $O(kd/n)$ where $kd \gtrapprox n^4$, from here the approximation that $1/n$ is the parameter size of a PHM layer. The

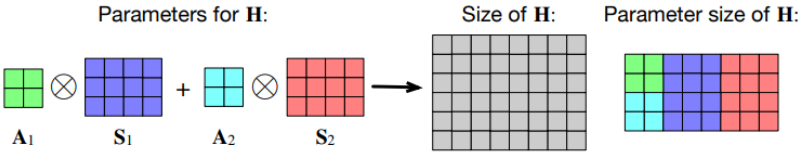


**Figure 3.3.** Illustration of the PHM layer [42]

reusing of the parameters from $A_i$ and $S_i$ reduces their numbers, and it saves time in training phase. The way PHM layers subsume the hypercomplex multiplication, for instance the Hamilton product, can be viewed in practice if we think of this example: a Hamilton product between two quaternions $Q$ and $P$:

$$\begin{bmatrix} Q_r & -Q_x & -Q_y & -Q_z \\ Q_x & Q_r & -Q_z & Q_y \\ Q_y & Q_z & Q_r & -Q_x \\ Q_z & -Q_y & Q_x & Q_r \end{bmatrix} \begin{bmatrix} P_r \\ P_x \\ P_y \\ P_z \end{bmatrix}$$

In order to use the Hamilton product all inputs are splitted into 4 $P_r, P_x, P_y, P_z$, then the components of the matrix are used as block matrices, the shape is aligned with $d$ that is the input length and $k$ is the output length. When using $n = 4$ the

PHM learns to simulate the Hamilton product of two quaternions. The $A_1, .., A_4$ matrices can be seen as the matrices composed by -1,0-1 that the Hamilton product uses as seen in 3.12. This is clearly visible in 3.15

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}_1} \underbrace{[Q_r]}_{\mathbf{s}_1} + \underbrace{\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A}_2} \otimes \underbrace{[Q_x]}_{\mathbf{S}_2} + \underbrace{\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_3} \underbrace{[Q_y]}_{\mathbf{S}_3} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_4} \underbrace{\mathbf{Q}_z}_{\mathbf{S_4}} \tag{3.15}$$

When $n = 1$ the PHM layer can also subsumes the matrix multiplication in the real field. If we take $H$ in this case is composed by $\mathbf{H} = \mathbf{A}_1 \otimes \mathbf{S}_1 = a\mathbf{S}_1$. When the network learns $a$ and $S$ separately we can drop the scalar $a$ and we have to learn only the weight matrix so this results in a classical fully connected layer matrix.

#### 3.2.1.2 Parameterized Hypercomplex Convolutional Layer

In [13] for the first time is proposed a paremeterize hypercomplex convolutional layer (PHC). As previously seen those layer are formally defined as

$$y = \text{PHC}(\mathbf{x}) = \mathbf{H} * \mathbf{x} + \mathbf{b},$$

with $\mathbf{H} \in \mathbb{R}^{s \times d \times k \times k}$, $s$ is the input dimension of the layer, while $d$ is the output and $k$ is the filter size. Similarly to 3.14 we define $H$ as

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{A}_i \otimes \mathbf{F}_i,$$

here $\mathbf{A}_i \in \mathbb{R}^{n \times n}$ remains of the same dimensions and represents the description of the algebra rules while $F_i \in \mathbb{R}^{\frac{s}{n} \times \frac{d}{n} \times k \times k}$ changes its dimensions and represents the batch of filters. Those matrices are learnt at training time, and have a reduction in the number of parameters, following the same reasoning of the linear layer done in the previous section. If we think that the degree of freedom of those matrices are $n^3$ and $sdk^2/n$. Because we are considering deep layers where $s$ and $d$ are bigger values with respect to k hence $sdk^2 \gg n^3$, so the PHC can be approximated to have

$o(sdk^2/n)$ degrees of freedom, thus the number of parameters is reduced by $1/n$ with respect to a standard convolutional layer. PHC layers share the weights among

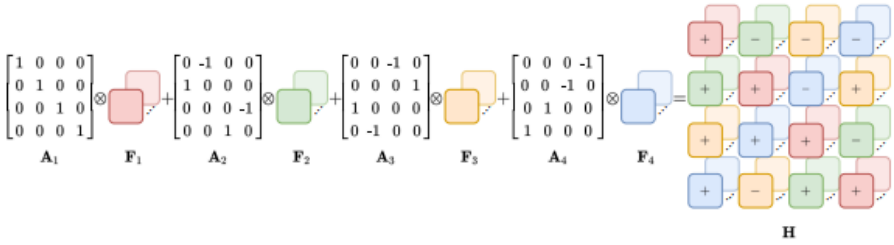


**Figure 3.4.** Example of the sum of Kronecker products with n=4 [13]

different channels, in this way the latent intra-channels relations are captured. As demonstrated in [13] the layers successfully learns how to subsume the Hamilton product, in particular the matrix $\mathbb{A}$ and so the Hamilton convolution. It's also reported that real convolution, with $n = 1$, can be represented by these layers.

# Chapter 4

# Proposed approaches

In the previous chapters we have seen generative adversarial network and more specifically the problem of IMT, then we presented hypercomplex deep learning in its newest approaches. In this chapter we will see how to put together those two: we will present Hypercomplex generative models that do Image to image translation and image modality translation. We will show how to model these huge architectures, using hypercomplex approaches and correspondingly the introduction of the novelties introduced: quaternion instance normalization and parameterized hypercomplex transposed convolution layers. Also an important analysis on weights initialization is done to find the best suitable configuration for those hypercomplex models.

## 4.1 Hypercomplex StarGAN v2

StarGAN v2 has been introduced in [4] and aims to resolve the problems of a scalable approach and a diverse image synthesis at the same time. They replace the concept of domain label with the style code that represent one of more styles from a specific domain. This is done using a mapping network or a style encoder, more specifically: the mapping network learns to transform a random noise to a style code, on the other hand the style encoder network produces also a style code but extracted from a reference image given in input. We will present in the next section the quaternion architecture of this network and also the approaches proposed to solve issues. The components of the quaternion architecture are replaced by PHM layers whenever we

will do the experiments for that model in the next chapter.

### 4.1.1 Architecture

To the purpose of visualizing better the architecture proposed, we will present some common blocks in the architecture and then they will be used inside the models. In figure 4.1 there are the residual blocks used in the architecture, they are skip-

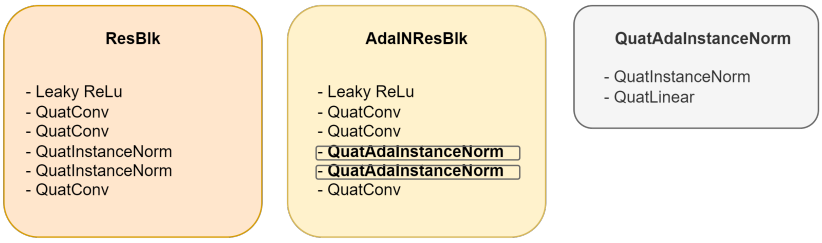


**Figure 4.1.** StarGAN v2 Residual blocks

connection blocks that learn residual function with reference to the layer inputs. Each residual is composed by different quaternion convolutions used to do upsampling or downsampling depending on the zone where they are used inside the models, they made use also of quaternion instance normalizations and uses as activation function the Leaky ReLU. The Adaptive instance normalization is used widely in this model and for the majority of the time it's used inside a residual block that we call Adaptive Instance residual block, composed by different quaternion convolutions.

#### 4.1.1.1 Generator

It takes as input an image and a style code of a domain that can be provided by the mapping network or the style encoder $G(x, s)$. Before injecting the code into the generator it is normalized using the adaptive instance normalization [18]. The complete architecture is showed in 4.2, there are four downsampling blocks, intermediate blocks and upsampling blocks. There is a particular thing to notice, the usage of a support network when using datasets of human faces, like Celeba HQ [20]. When we are using this support network, we will not use shortcuts of residual blocks when we are upsamplng instead skip connections with the heatmap provided by [37], basically as the figure says it will provide the shape aligned of the input

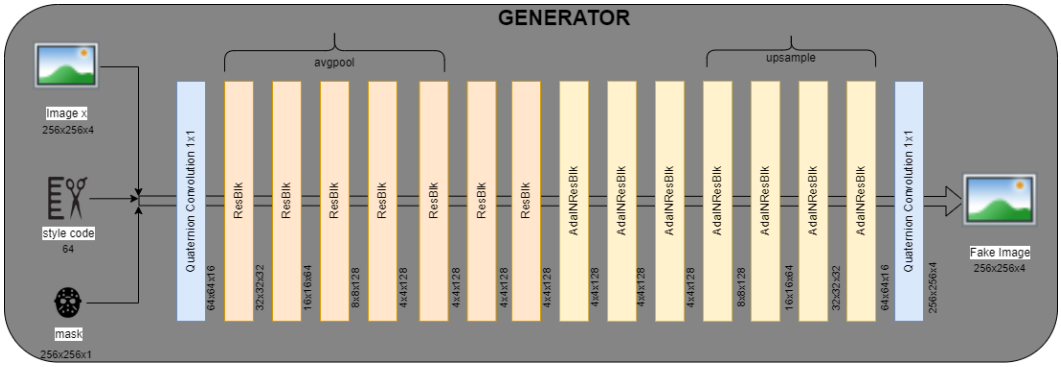


**Figure 4.2.** StarGAN v2 generator

face.

#### 4.1.1.2 Mapping network

This model provides to the generator a style code $s = F_y(z)$ given a latent code $z$. Depending on the number of domains, this network has different output branches, in this way it can provide a code for each domain. The output codes results in a diverse interpretation of the domain each time because of the randomness of the input code. As we can see from figure 4.2 this network make a large use of quaternion linear

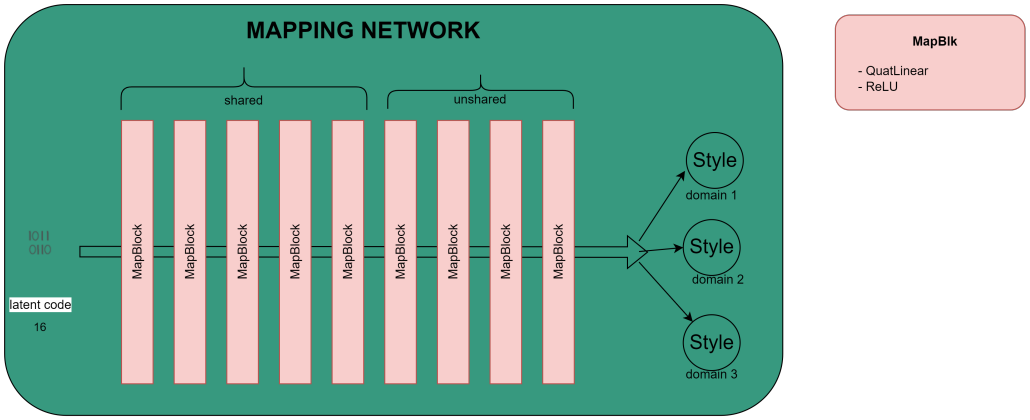


**Figure 4.3.** StarGAN v2 Mapping network

layers and uses as activation function the ReLU. The network is composed by the MLP, four fully connected layers that have shared weights between the domains. The latent code used is of dimension 16 while the hidden dimension is of 512 and the style code in output has 64 as dimension.

#### 4.1.1.3 Style encoder

This network also provides style codes, but it will start from an input image, hence $s = E_y(x)$. In this model there are $k$ output branches depending once again by the number of domains, there are residual blocks shared among all domains, and at the end we have the style block composed by a convolution and two quaternion linear layer, the output shape of this network is 64.

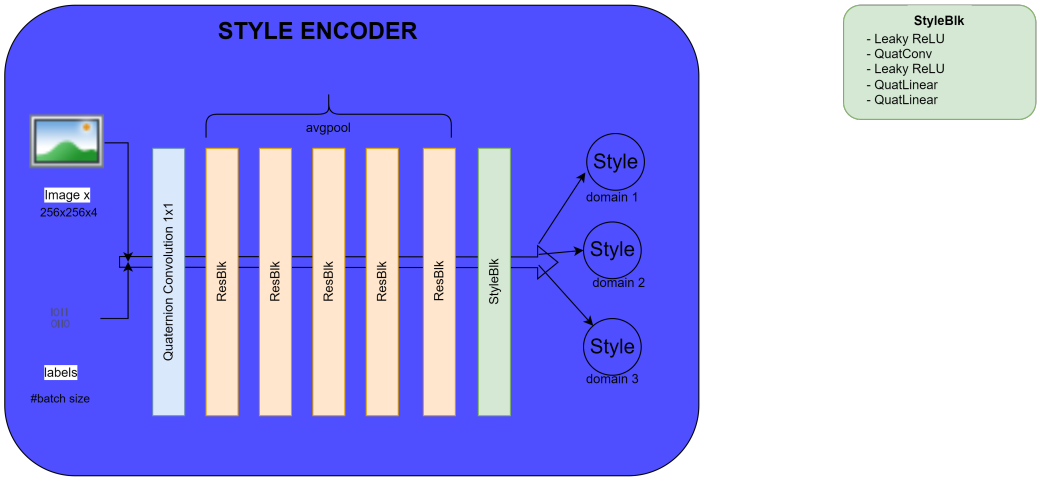


**Figure 4.4.** StarGAN v2 Style Encoder

#### 4.1.1.4 Discriminator

It is a multi-task discriminator, for each domain it has a branch that is a binary classifier and learns to distinguish if an image is real or fake depending on its domain. Also this model uses residual blocks, for instance five. The final discriminator block has an additional quaternion fully connected layer with respect to the original architecture. Since we are using quaternions that have at least 4 channels, after the last convolution we solve the size problem with this dense layer.

### 4.1.2 Training Objectives

Given the number of tasks that this model has to handle it comes with different loss functions, each points to solve a particular problem:

- **Adversarial objective:** a random domain $\tilde{y} \in \mathcal{Y}$ and a random latent code

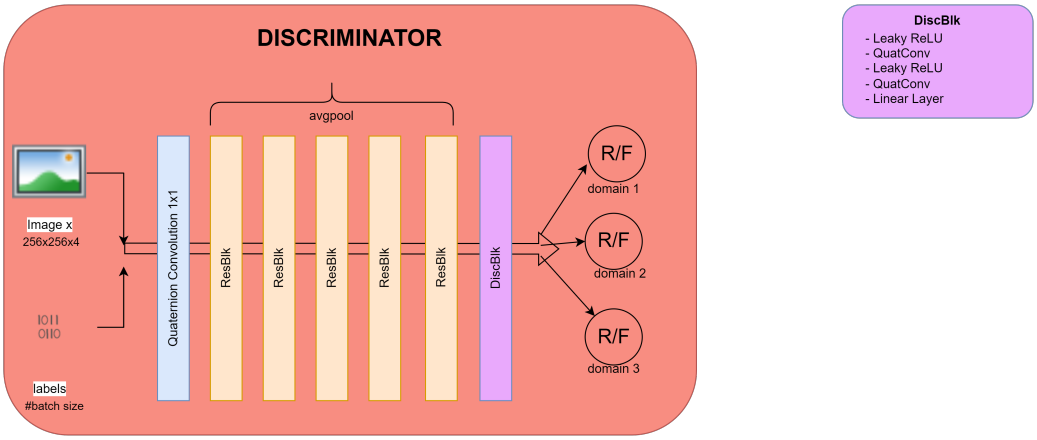


**Figure 4.5.** StarGAN v2 Discriminator

$z \in \mathcal{Z}$ are sampled during the training phase. The mapping network is in charge of generating a style code $\widetilde{\mathbf{s}} = F_{\widetilde{y}}(\mathbf{z})$, which is passed to the generator together with an image $x$, $G(\mathbf{x}, \widetilde{\mathbf{s}})$.

We are using an adversarial loss, $D$ is the discriminator; in this way the mapping network learns to provide the right style codes for the domain we are analyzing and the generator learns to use correctly this style code to provide good fakes to the discriminator.

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathbf{x},y}\left[\log D_y(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x},\widetilde{y},\mathbf{z}}\left[\log\left(1 - D_{\widetilde{y}}(G(\mathbf{x}, \widetilde{\mathbf{s}}))\right)\right] \tag{4.1}$$

- **Style reconstruction:** this is used to force the generator to use the style code $\widetilde{\mathbf{s}}$. This loss function is similar to approaches [49] where there are multiple encoders and the objective is to learn a mapping from an image to its latent code.

$$\mathcal{L}_{sty} = \mathbb{E}_{\mathbf{x},\widetilde{y},\mathbf{z}}\left[\left\|\widetilde{\mathbf{s}} - E_{\widetilde{y}}(G(\mathbf{x}, \widetilde{\mathbf{s}}))\right\|_1\right] \tag{4.2}$$

- **Style diversification:** this aim to force the generator to produce synthetic images that are different. The generator is regularized following the results obtained in [24].

$$\mathcal{L}_{ds} = \mathbb{E}_{\mathbf{x},\widetilde{y},\mathbf{z}_1,\mathbf{z}_2}\left[\left\|G\left(\mathbf{x}, \widetilde{\mathbf{s}}_1\right) - G\left(\mathbf{x}, \widetilde{\mathbf{s}}_2\right)\right\|_1\right] \tag{4.3}$$

The $z_1$ and $z_2$ are generated from the mapping network starting from two random latent codes.

- **Preserving source characteristics:** the generated images have to preserves some characteristics from the sources, for example the pose. This is ensured by this cycle consistency loss. The generator is learning to preserve some of the characteristics and also at the same time changing its style.

$$\mathcal{L}_{cyc} = \mathbb{E}_{\mathbf{x},y,\widetilde{y},\mathbf{z}} \left[ \|\mathbf{x} - G(G(\mathbf{x}, \widetilde{\mathbf{s}}), \hat{\mathbf{s}})\|_1 \right] \tag{4.4}$$

The previous loss functions are all reguralized using r1 regularization [25]. The final objective function uses some weights, the lambdas in 4.5, as hyperparameters to tune it during training. The model is also trained using the style encoder in the same manner it does when it uses the mapping network, as said previously, but the style codes are provided by the style network.

$$\min_{G,F,E} \max_{D} \mathcal{L}_{adv} + \lambda_{sty}\mathcal{L}_{sty} - \lambda_{ds}\mathcal{L}_{ds} + \lambda_{cyc}\mathcal{L}_{cyc} \tag{4.5}$$

### 4.1.3 Quaternion Instance Normalization

Starting from the idea that we are in a different domain, we may want to explore the usage of different normalizations. Following the path of changing the real valued batch normalization by a quaternion one, as seen in section 3.1.2.4, we decided to introduce, for the first time in literature, the quaternion instance normalization.
The idea of instance normalization was introduced for the first time by Ulyanov in [35] and results to be effective with respect to the classical batch normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tim}, \quad \sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tim} - \mu_{ti})^2. \tag{4.6}$$

The key difference lays in the fact that $\mu(x)$ and $\sigma(x)$ are computed across spatial dimensions independently for each channel and each sample, this difference is clearly visible in 4.6. At test time the instance normalization layers are applied without using population statistics, that happens instead in the batch one. The way in which

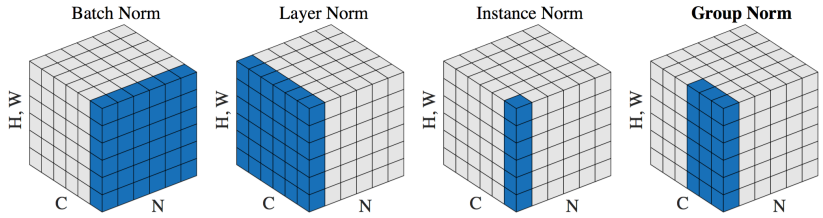


**Figure 4.6.** Different types of normalizations [38]

the quantities are computed changes, the mean computed is a quaternion value and not a scalar, for instance the mean is computed as:

$$\mu_q(q) = \frac{1}{HW}\sum_{t=1}^{H}\sum_{t=1}^{W}(q_t) = \bar{r} \ \ | \ \bar{x}\boldsymbol{i} \ \ | \ \bar{y}\boldsymbol{j} \ \ | \ \ \bar{z}\boldsymbol{k} \tag{4.7}$$

the components of $\mu_q$ are stacked together after the mean is computed for each separated channel. The covariance is, as the variance, calculated across the mean of sum of the squared components of the quaternionic mean:

$$\sigma_q(q) = \frac{1}{HW}\sum_{t=1}^{H}\sum_{t=1}^{W}(\Delta r^2 + \Delta x^2 + \Delta y^2 + \Delta z^2)$$

From here the final equation that formalize the quaternion instance normalization, interesting to see that the four components share the same variance but using their own mean.

$$\begin{aligned} QIN(q_i) &= \gamma(\frac{q_i - \mu_q(q)}{\sqrt{\sigma_q(q) + \epsilon}}) + \beta \\ &= \gamma(\frac{r_i - \overline{r}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}) + \gamma(\frac{x_i - \overline{x}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}) + \gamma(\frac{y_i - \overline{y}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}) + \gamma(\frac{z_i - \overline{z}_i(q)}{\sqrt{\sigma_q(q) + \epsilon}}) + \\ &+ (\beta_r + \beta_x + \beta_y + \beta_z) \end{aligned}$$

In the results section we will see how this introduction can change the networks in a better way.

Consequently we can consider to extend the same reasoning to the adaptive instance

normalization used in StarGAN v2:

$$\text{AdaIN}(x, y) = \sigma(y) \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y). \tag{4.8}$$

This method aligns the mean and variance of $x$, input images, with the style features of those images $y$, trying to match them. This normalization does not use learnable affine parameters because they are computed from the style in input in an adaptive way.

This normalization is concretely realised using a quaternion linear layer and the quaternion instance normalization mentioned above, in this way the model learns adaptively the parameters for this normalization. Considering the example of an image that has brushstrokes of a particular style, the style image associated will have high activation for this feature, in the same manner the AdaIN preserve the spatial structure but maintain the high activation for this feature. This layer not add computational cost (with respect to Instance normalization) and transfers features statistics successfully.

### 4.1.4 Weights Initializations

A crucial point in a neural network is the weights initialization of the layers. One could think to initialize the weights randomly but the values can be close to zero in this way the gradients in the initial layer will vanish, this phenomena is called vanishing gradients. On the other side when the gradients are greater than 1 they can explode and the network will not train at all. In the original work of StarGAN v2 the initialization is done using the *he* as reported in [14] using a normal distribution, meaning that the values will be sampled from $\mathcal{N}(0, \sigma^2)$ where

$$\sigma = \frac{\text{gain}}{\sqrt{\text{fan\_mode}}}.$$

The "fan mode" could be "fan in" that preserves the magnitude of the variance of the weights in the forward pass, while "fan out" preserves the magnitudes in the backward pass.

When using PHM layers a generic *xavier* initialization [9] is used for the weight

matrices $A$ and $S$. This initialization uses a uniform distribution, so the values will be sampled from $\mathcal{U}(-a, a)$ where gain is a generic scaling factors.

$$a = \text{gain} \times \sqrt{\frac{6}{\text{fan\_in} + \text{fan\_out}}}$$

In the quaternion version of this StarGAN v2 we are using *he* when there are linear layers, and *Glorot* when it comes to convolutions.

The need of a different initialization comes when we discovered that the PHM StarGAN v2 does not train at all when using the canonical initialization for the weight matrices. This lead to a deep analysis of the problem and the successful discovery of the particular initialization that could benefit this model. In particular we decided to use an approach starting from the quaternion initial phase. The $A$ matrix when $n = 4$ is initialized using this matrix stacked together in a tensor in 4.9:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}_1} \underbrace{\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbf{A}_2} \underbrace{\begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_3} \underbrace{\begin{bmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{A}_4} \tag{4.9}$$

While for the $S$ matrix we followed the initialization done in [30] that we can divide in two steps:

1. A purely imaginary quaternion $q_{imag}$ is generated following an uniform distribution in the interval [0, 1]

2. The imaginary unit is then normalized, denoted as $\overline{q}_{imag}$

$\phi$ is a random variable wrt the variance of the quaternion weight and the initialization criterion (He for linear layers and Glorot for convolutional layers). The weights are

so initialized in this way, then they are joint together to create the $S$ matrix.

$$\begin{aligned} w_{\mathbf{r}} &= \phi * \overline{q}_{imag\mathbf{r}} * \cos(\theta), \\ w_{\mathbf{i}} &= \phi * \overline{q}_{imag\mathbf{i}} * \sin(\theta), \\ w_{\mathbf{j}} &= \phi * \overline{q}_{imag\mathbf{j}} * \sin(\theta), \\ w_{\mathbf{k}} &= \phi * \overline{q}_{imag\mathbf{k}} * \sin(\theta). \end{aligned} \tag{4.10}$$

When $n = 3$ the shape of the matrix $A$ changes too, so we decided to generate random matrices like the 4.9, with numbers in the interval [-1,1], but with a shape coherent with the $n$ chosen. The same holds for the $S$, here we are taking only the first three component of the weights generated in the same way of 4.10.

## 4.2 Hypercomplex TarGAN

Image to image translation is a task that could be really helpful in medical applications. Nowadays paired multi-modality images can provide complementary information to help medical staff in order to make decisions based on multiple factors. The problems are that those images are rare to find because of the time or due to the cost of multiple radiation tests. In this way image-to-image translation found a fertile area where there is a urgent need of those images. In literature there are multiple works on that topic but all of these concentrate on the translation task of a complete image and not concentrating on a specific area called: *Region of Interest (ROI)*. In many cases the translated images has blurry part on that ROI or also deformed area.

This is where TarGAN [2] intervenes: it does multi-modality medical image translation and enhance the quality of the target area of interest, the model does at the same time both the tasks. The problem of the current approaches are similar to the general problems of image to image translation networks, so the scalability issues that grows with the number of domains [41] and also the existent methods rely on paired data that are difficult to find in a real world medical scenario [39].

### 4.2.1 Architecture

The network is composed by different blocks, in 4.7 we could see the principal ones. The following architecture uses PHM layers, but the same holds also for a classical quaternion approach. The **ConvBlock** is composed by a parameterized hypercomplex convolution, the quaternion instance normalization, and the leaky ReLU as activation function. The **UPConvBlock** instead is made of a interpolation using nearest neighbor, a convolution and a quaternion instance normalization, it will use the leaky ReLU as activation function as well.

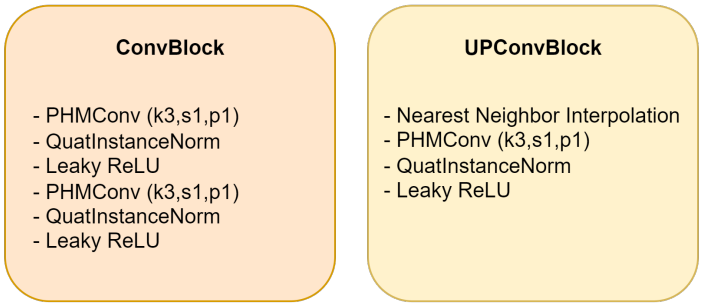


**Figure 4.7.** TarGAN blocks

#### 4.2.1.1 Generator

It does both segmentation and translation. The model takes as input an image $x_s$, the target area associated $r_s$ and the output target modality $t$.

$$G(x_s, r_s, t) \rightarrow (x_t, r_t)$$

This network has an encoder in input, some shared middle blocks, and a decoder to output the final results, this is showed in 4.8. There are two pairs of this encoder-decoder combination one input is the entire image $x_s$ the other includes only the target area $r_s$, the sharing of the middle block is enforcing G on focusing on the ROI when translating the image.

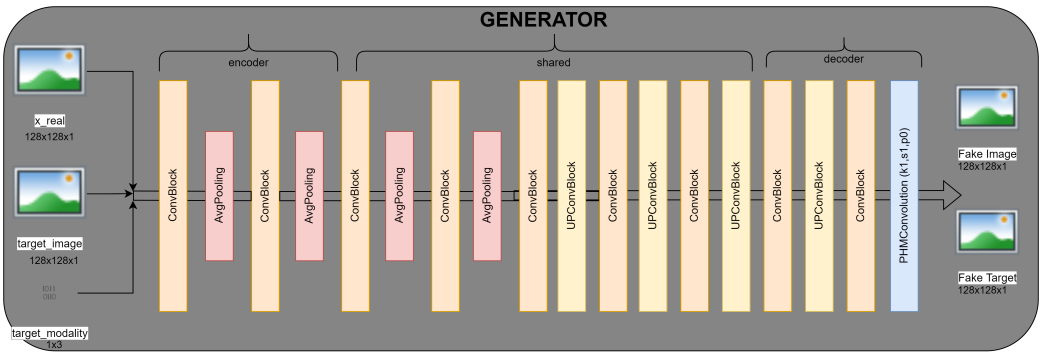


**Figure 4.8.** TarGAN Generator

#### 4.2.1.2 Shape Controller

$S$ takes as input a fake image $x_t$ or a fake target image $r_t$ and generates binary masks that represent the foreground area of the generated images. As shown in 4.9, it uses the ConvBlock presented at the start of this section and changes the operation of pooling between max and up, where up is a transposed convolution used to do upsampling.

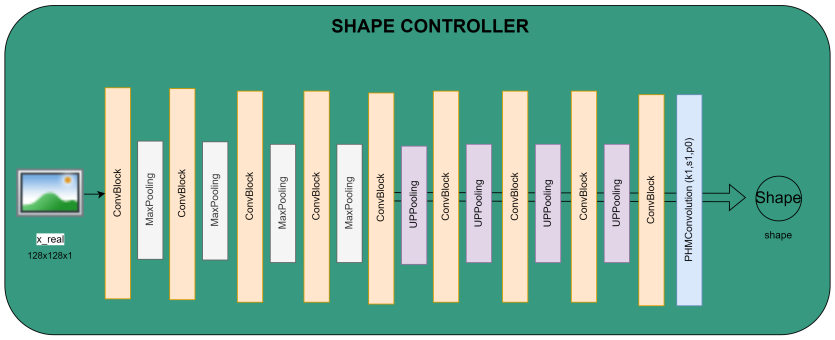


**Figure 4.9.** TarGAN Shape Controller

#### 4.2.1.3 Discriminators

We use two discriminators due to the two different output the generator provides. The same architecture is used and it is showed in 4.10

- **Discriminator images** $D_x$ it establish if a generated sample is real or not and also from which modality it has generated from.
- **Discriminator targets** $D_r$ it establish if a target area is real or not and like

the image discriminator from what domain it comes from.

The difference in the two models is in the final layer that is a convolution, with kernel size equals to 3,stride is 1 and one as padding, when we are speaking of $D_x$. While the output of the $D_x$ is a convolution with dimension depending on the number of target modalities, and a kernel size depending on the image size.

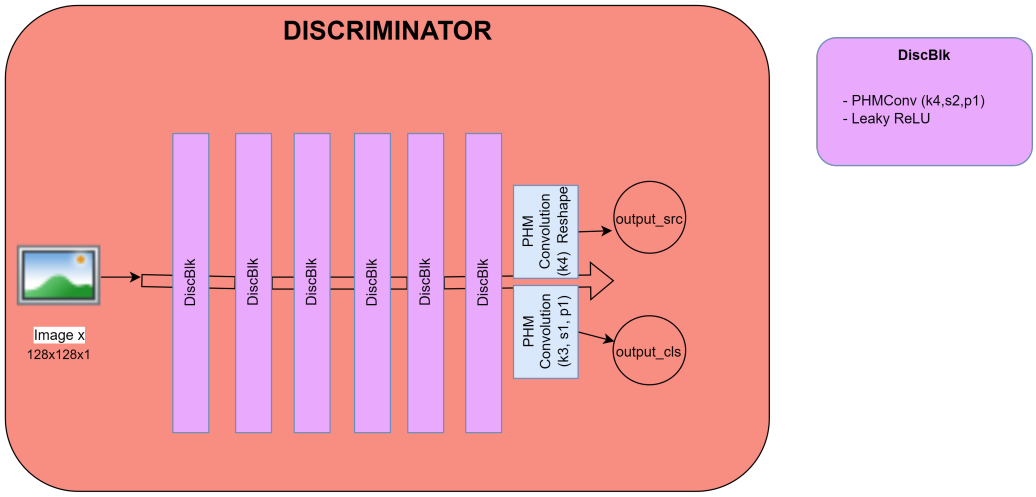


**Figure 4.10.** TarGAN Discriminator

### 4.2.2 Training Objectives

- **Adversarial loss:** it is the canonical loss to train adversarial networks.

$$\begin{aligned} \mathcal{L}_{adv\_x} &= \mathbb{E}_{x_s}\left[\log D_{src\_x}\left(x_s\right)\right] + \mathbb{E}_{x_t}\left[\log\left(1 - D_{src\_x}\left(x_t\right)\right)\right] \\ \mathcal{L}_{adv\_r} &= \mathbb{E}_{r_s}\left[\log D_{src\_r}\left(r_s\right)\right] + \mathbb{E}_{r_t}\left[\log\left(1 - D_{src\_r}\left(r_t\right)\right)\right] \end{aligned} \tag{4.11}$$

- **Modality classification loss:** this is used to force the target modality and applied on $G$, $D_x$ and $D_r$. $\mathcal{L}^r_{cls\_x}$ refers to the loss of real images, thus used for the discriminators,

$$\begin{aligned} \mathcal{L}^r_{cls\_x} &= \mathbb{E}_{x_s,s}\left[-\log D_{cls\_x}\left(s \mid x_s\right)\right] + \lambda_u \mathbb{E}_{x_t,s'}\left[-\log D_{cls\_x}\left(s' \mid x_t\right)\right] \\ \mathcal{L}^r_{cls\_r} &= \mathbb{E}_{r_s,s}\left[-\log D_{cls\_r}\left(s \mid r_s\right)\right] + \lambda_u \mathbb{E}_{r_t,s'}\left[-\log D_{cls\_r}\left(s' \mid r_t\right)\right] \end{aligned} \tag{4.12}$$

while $\mathcal{L}^{f}_{cls\_x}$ to the fake one and used on the generator $G$.

$$\mathcal{L}^{f}_{cls\_x} = \mathbb{E}_{x_t,t}\left[-\log D_{cls\_x}\left(t \mid x_t\right)\right], \quad \mathcal{L}^{f}_{cls\_r} = \mathbb{E}_{r_t,t}\left[-\log D_{cls\_r}\left(t \mid r_t\right)\right]$$

- **Shape consistency loss:** this is used because the untraceable constraint [47] interfere sometimes with the anatomy shape of the structures in the generated samples, so the shape loss contains this behaviour.

$$\mathcal{L}_{\text{shape }\_x} = \mathbb{E}_{x_t,b^x}\left[\left\|b^x - S\left(x_t\right)\right\|_2^2\right], \quad \mathcal{L}_{\text{shape }\_r} = \mathbb{E}_{r_t,b^r}\left[\left\|b^r - S\left(r_t\right)\right\|_2^2\right] \tag{4.13}$$

- **Reconstruction loss:** this is used also in the StarGAN v2 that took inspiration from [48], it aim to preserve the original characteristics of the source images.

$$\mathcal{L}_{rec\_x} = \mathbb{E}_{x_s,x'_s}\left[\left\|x_s - x'_s\right\|_1\right], \mathcal{L}_{rec\_r} = \mathbb{E}_{r_s,r'_s}\left[\left\|r_s - r'_s\right\|_1\right] \tag{4.14}$$

- **Crossing loss:** this is used to force the generator to pay attention to the target area when generating the translated image. $y$ is the target area label of $x_s$.

$$\mathcal{L}_{\text{cross}} = \mathbb{E}_{x_t,r_t,y}\left[\left\|x_t \cdot y - r_t\right\|_1\right], \tag{4.15}$$

The complete objective function is reported below 4.16, there are also lambdas to controls the weights of the loss functions.

$$\begin{gathered}
\mathcal{L}_{D_{(x/r)}} = -\mathcal{L}_{adv_(x/r)} + \lambda^{r}_{cls}\mathcal{L}^{r}_{cls_(x/r)} \\
\mathcal{L}_G = \mathcal{L}_{adv_(x/r)} + \lambda^{f}_{cls}\mathcal{L}^{f}_{cls_(x/r)} + \lambda_{rec}\mathcal{L}_{rec_(x/r)} + \lambda_{cross}\mathcal{L}_{crossing}, \\
\mathcal{L}_{G,S} = \mathcal{L}_{shape_(x/r)},
\end{gathered} \tag{4.16}$$

### 4.2.3 Depthwise concatenation for biomedical images

In the previous section we have seen how the quaternions exploit the intra-channel relations when we are dealing with multi channels objects like images, we have taken

for granted that an image is composed by the three components RGB. When we are treating biomedical images, we can easily notice that the images have only one channel, because they are grayscale images. This is clearly a problem for our work in the quaternion domain where until now the effectiveness of the proposed approaches sees different channels for the images.

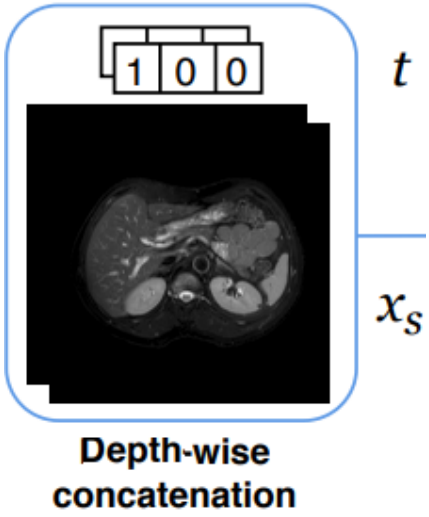


**Figure 4.11.** TarGAN depthwise concatenation

There are different solutions and opportunities that this situation brings up to. One possibility of still using the quaternions lays in the fact that the generator takes as input also the target modality, the occasion was that the modality in this case is a three dimensional vector because of the three modality present in the dataset we are investing to. So the solution to use the quaternion in this domain is the depthwise concatenation of the input image with the modality it has to be translated to, in this way we will have an input that respect the constraint given by the quaternion of having 4 channels.

Subsequently to this solution for the generator, a similar problem lays in the other models of the architecture, but with the difference that a discriminator not uses the target modality to distinguish whether an image is real or fake and the same for the Shape Controller. This led to a simple but effective solution, to replicate the only channels of the biomedical image in other three, this is an approach that is commonly used to deal with black and white images.

### 4.2.4 PHM Transposed convolution

A normal convolution has kernels that moves through the input's layers and produces a new output, in the image 4.12 is visible blue squares are inputs, and green squares are outputs.

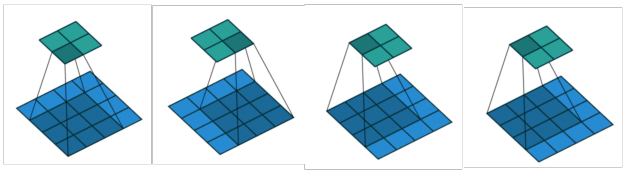

**Figure 4.12.** Convolution visualized with k=3 and a 4x4 input, producing a 2x2 output [7]

In order to do upsampling in the Shape controller model, there are transposed convolutions, in opposite to the convolutions presents that instead do a downsampling by definition. Basically the input and the output dimensions switch places, resulting in a larger output with respect to the input.

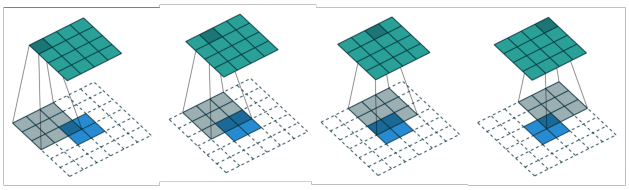

**Figure 4.13.** Partial visualization of a transposed convolution, with k=3 and a 2x2 input, producing a 4x4 output.

There are some particularities with those layers, for example the padding reduce the sizes of the output as it is removed from the input, or when using the strides they affects the input instead of the output. Since the novelty of this approach with parameterized hypercomplex multiplications, not all the canonical layer used in CNN are yet implemented and tested. This is why we are introducing this new approach also to the transposed convolution, the dimensions of the matrix $F$ are switched, $F_i \in \mathbb{R}^{\frac{d}{n} \times \frac{s}{n} \times k \times k}$. A partial visualization of this process is showed in 4.13.

### 4.2.5 Mixed approach

To explore new solutions to the problem of using hypercomplex methods on biomedical data, we developed a mixed approach. There are some limitation maybe in the

method of the depthwise concatenation because, as said before, the output will be of only one channel so the discriminator has to replicate the others missing. To overcame this, we decided to adopt a mixed strategy: the input layer and the output layer of the discriminator and the shape network will be real convolutional layers. In this way we are not generating replicated channels to keep the restriction on the four channels needed for the quaternion approach and for the depthwise approach also for the PHM case. Another thing that we have done is to not use hypercomplex layers for the shared layers of the generator showed in 4.14. So the encoders and the decoders remains hypercomplex, PHM or quaternion, while the shared blocks will be real, this is to not make confusion between information when we are mixing the weights of different flows, in this case one will translate and another segment the original image.

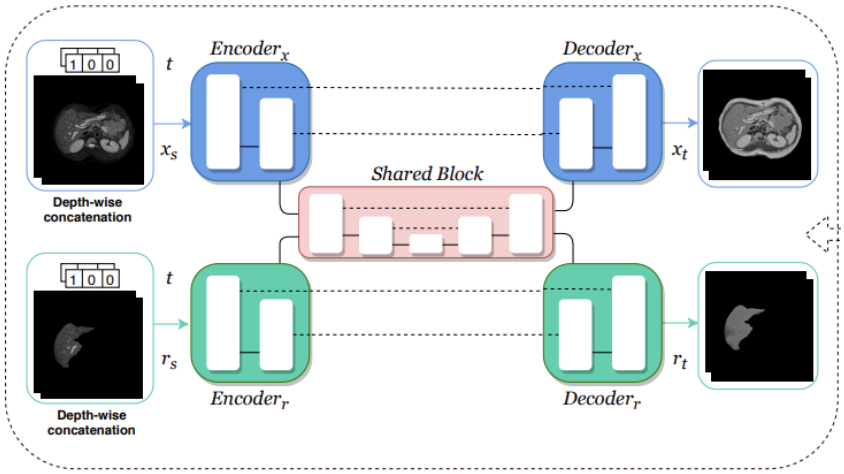


**Figure 4.14.** TarGAN Generator with overview on shared layers [2]

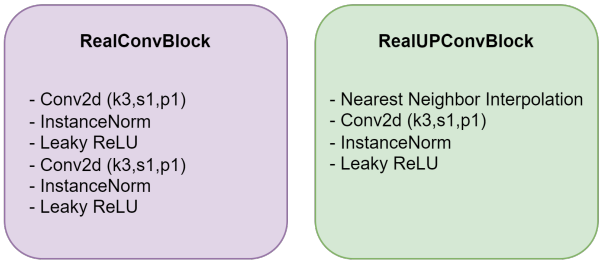


**Figure 4.15.** TarGAN real blocks

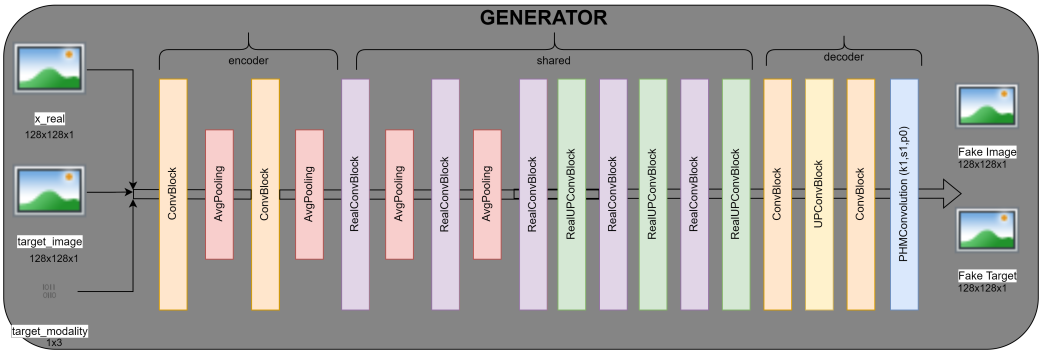


**Figure 4.16.** TarGAN mixed Generator

The generator will be composed as shown in 4.16, the blocks used are the ones we saw before in 4.7 and the real presented here can be seen in 4.15.

# Chapter 5

# Experimental Results

In this chapter we will describe datasets and discuss scenarios of all the achieved experiments. Experiments are considered on different tasks and datasets, an important thing to notice is the fact that the samples contained in the dataset are multidimensional and correlated, which is one of the motivations of this work in hypercomplex domains. It is done a comparison with respect to the existing methods, using the same metrics as in the original works, we will evaluate the translation task using as metrics the Frechet Inception Distance and the LPIPS, while the segmentation task with different metrics like DICE, Segmentation score and RAVD. All these metrics will be later explained. Then we will analyze the results obtained with StarGANv2, their pros and their cons, and the results obtained with TarGAN in the biomedical field, so also for the segmentation.

## 5.1 Datasets

In this section we will present the datasets used in the experiment section, those datasets are the one used in the original works, so the comparisons will be transparent and fair.

### 5.1.1 Celeba HQ

Introduced in [20], this dataset is a high quality version of CELEBA. The original dataset has a variety of images, but with different resolutions and different targets

(for example some photos have crowds of people on the back).
So to build the dataset some preprocessing is done:

- Using neural networks specialized in removing artifacts in jpeg
- After this a 4x super resolution has been done on the dataset to propose better quality images.
- An oriented crop rectangle based on the facial landmark is computed and then transformed in 4096x4096 rectangle and scaled to 1024x1024 helped by a box filter.

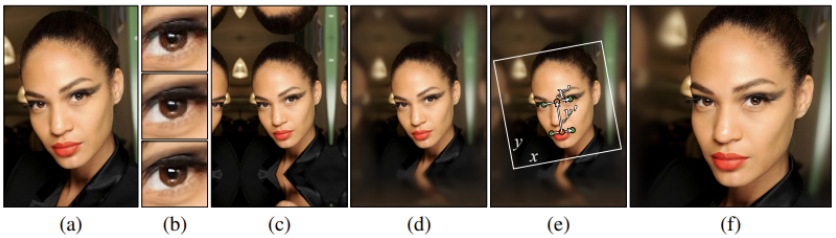


**Figure 5.1.** Creation of the CELEBA-HQ dataset

After this preprocessing only 30000 photos will be selected, considered the best on a quality check done using a frequency based quality metric, this will penalizes blurry images and images that are not visually good.

### 5.1.2 Animal Faces HQ

This dataset is composed by 15000 high quality images at a 512x512 resolution, first introduced in [4] animal dataset. It has three categories: cat dog and wildlife animals, it is balanced because each category has 5000 images. Each category has at least eight different breads. The preprocessing done is only an alignment, to center the faces, of the original photos taken by Flickr and Pixabay websits.

### 5.1.3 Chaos 2019

This dataset contains CT and MR scans from unpaired abdominal image series, the ground truth is generated with the auxiliary participation of multiple experts annotations, introduced for the first time in [21]. The data comes from 80 patients

**Figure 5.2.** AFHQ dataset samples

from the Department of Radiology of Dokuz in Turkey, half of them have done only a single CT scan while the other went through MR scans. All of the scans are from healthy patients, meaning that no tumor or metastasis is present in this dataset.

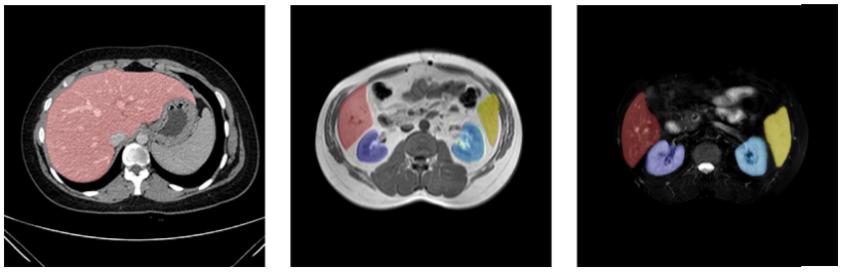

**Figure 5.3.** Chaos dataset. Liver segmentation in red, right kidney dark blue, left kidney light bllue and spleen in yellow

- CT volumes were acquired using a contrast agent injection, in this way the liver parenchyma is enhanced. The task of segmentation is done only on the liver for this modality, because it is widely used for that.
- MRIs samples are divided in two different sequences T1 and T2:
  - There are 40 samples from **T1-Dual** in-phase and 40 samples from the opposite phase
  - There are 40 samples from **T2-SPIR**

The datasets were acquired on a 1.5T Philips MRI.

A brief explanation on what are the differences in our dataset modalities. MRI images are distinguished between T1 weighted and T2 weighted, the difference

between those two is in the timing of pulse radio frequency sequences. The pulse of T1 highlights the fat tissue while the T2 fat and water. The difference between those MRI and CT scans lays in the fact that CT is computed tomography done using X-rays while MRI uses radio waves.

## 5.2 Metrics

When evaluating GAN we are limited in the way we can evaluate a model that succeeds to generate good enough fake samples. We have to deal with different factors, we want both fidelity and diversity of the generated images. A basic approach is to use the pixel distances subtracting two images pixel values. In the next section we will see different methods to evaluate GANs and also to evaluate, in the case of TarGAN, the segmentation tasks.

### 5.2.1 Fréchet Inception Distance - FID

Fréchet Inception Distance (FID) captures the similarity of generated images to real ones better than the Inception Score as stated in [15], this work is the first one to introduce this metric. Frèchet distance measures the similarity between curves, taking into account the location and the ordering of points on the curves. In this metric is used to compute the distance between two distributions. It is given by $\mathrm{d}(\mathbf{X}, \mathbf{Y}) = (\mu_{\mathrm{X}} - \mu_{\mathrm{Y}})^2 + (\sigma_X - \sigma_{\mathrm{Y}})^2$ for univariate distribution.
This metric measure the feature distance between the real and the generated images. The Inception v3 model [33] is used pre-trained on the imagenet dataset, from here the "inception" inside the FID name. In particular the second-last pooling layer is used:

$$\mathrm{F\,ID} = ||\mu_X - \mu_Y, ||^2 - \mathrm{T\,r}(\sum_X + \sum_Y - 2 \quad \sum_X \sum_Y)$$

$X$ and $Y$ are real and fake embeddings from the inception, while $\mu_X$ and $\mu_Y$ are the magnitudes. $TR$ is the trace of the square matrix (the sum of elements on the main diagonal), while $\mathbf{\Sigma}_X$ and $\mathbf{\Sigma}_Y$ are the covariance matrix. The less is the FID value the better is the score of our network.

### 5.2.2 Learned Perceptual Image Patch Similarity - LPIPS

Another measure used in the GAN metric calculus is the LPIPS (Learned perceptual image patch similarity) [43]. This measure aim to substitute the perceptual metrics used today such as PSNR and SSIM, because fail to notice many nuances that human instead notice.

The paper used a new dataset (48400 entries) of human perceptual similarity judgements to evaluate deep features of different architectures. The differences between a reference image (real) and a distorted patch (fake) is calculated using:

$$d(x, x_0) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot \delta \left( \hat{y}^l_{hw} - \hat{y}^l_{0hw} \right) ||_2^2$$

We can extract features from arbitrary networks, we will extract features from l layers of AlexNet [22]. It measures the diversity of generated images (higher is better).

### 5.2.3 Dice coefficient - DICE

It was introduced by Dice in 1945 for ecological studies in [6]. Nowadays it is widely used for medical image segmentation, it quantifies how much two sets are similar averaging the size of their intersection:

$$DICE = \frac{2\,|V_{Ref} \cap V_{Seg}|}{|V_{Ref}| + |V_{Seg}|} \times 100$$

$V_{Ref}$ is the set of pixels identified as foreground in the ground truth image, and $V_{seg}$ is the same but for the segmented image. DICE is between 0 and 1 but for our task we will convert it to percentage, clearly the higher the value the higher is the correctness of the model.

### 5.2.4 Segmentation score - S-score

This measure was introduced in [44], to see how the shape of a segmentation behaves after the translation in another domain. For each fake segmented image, S-score is computed with the DICE between the generated segmented image and the ground

truth of the real image. The same reasoning holds for this measure, higher score means better matched shape.

$$Dice(SegNet(s), g)$$

Where $s$ is the generated image, and $g$ is the original image.

### 5.2.5 Relative absolute volume difference - RAVD

RAVD measures the similarity of two images, generally 3D images, it was introduced in [32], and used for medical image segmentation tasks evaluation.

$$RAVD = \frac{|| \mid V_{Seg} \mid - \mid V_{Ref} \mid ||}{\mid V_{Ref} \mid} \times 100.$$

The lowest is this value, the better is the segmentation, we will use the value in percentage.

## 5.3 StarGAN v2 experimental results

We tried different experiments according to the approach proposed in the previous chapter. The effectiveness of the solutions found will be clear after seeing the different metrics and results.
The experiment section of the Hypercomplex StarGAN v2 is divided in generic image modality translation and biomedical one, but the concepts and the final model it is the same.

### 5.3.1 Quaternion Instance Normalization

The first experiments uses the real instance normalization, but as said in the previous chapter, we want to exploit the fact that we are dealing with quaternions, to this aim we introduce the quaternion instance normalization. As we can see from both the loss functions 5.4 and the FID values, we are improving the model using this new normalization in the quaternion domain.

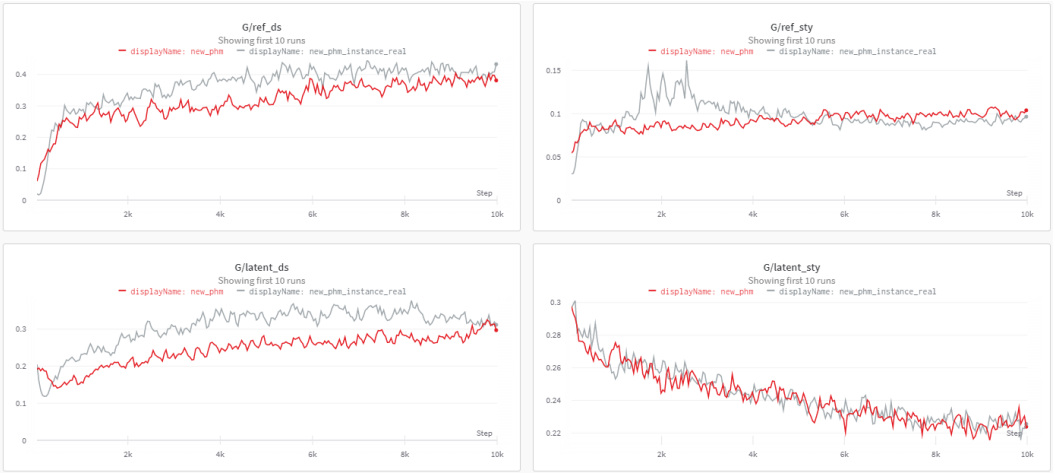


**Figure 5.4.** Generator loss functions compared, with quaternion instance normalization in red, with real instance normalization in grey

### 5.3.2 Discriminator with a dense layer

As mentioned before, one of the problem when dealing with hypercomplex domains is the restriction on the minimum number of channels a tensor have. In the discriminator architecture the last original layer uses a convolution, resulting in an output with the same number of domains as channels number, but since this is not always possible in hypercomplex domains, we introduced a fully connected layer (dense) as the last layer of the discriminator, notice that this layer is the real version. In figure 5.5 there is a comparison of the losses with the classical quaternion approach, but this hold also for the PHM approach, as we will see later in 5.2.

### 5.3.3 Weights initialization

Another issue that we analyzed is the one of the weights initialization. In particular for the PHM version of the network, we can use a canonical initialization for the matrix $S$ and $A$ of the PHM layers, instead we have used the quaternion one when dealing with quaternions, so with $n = 4$, and a revision of the latter when using $n = 3$.

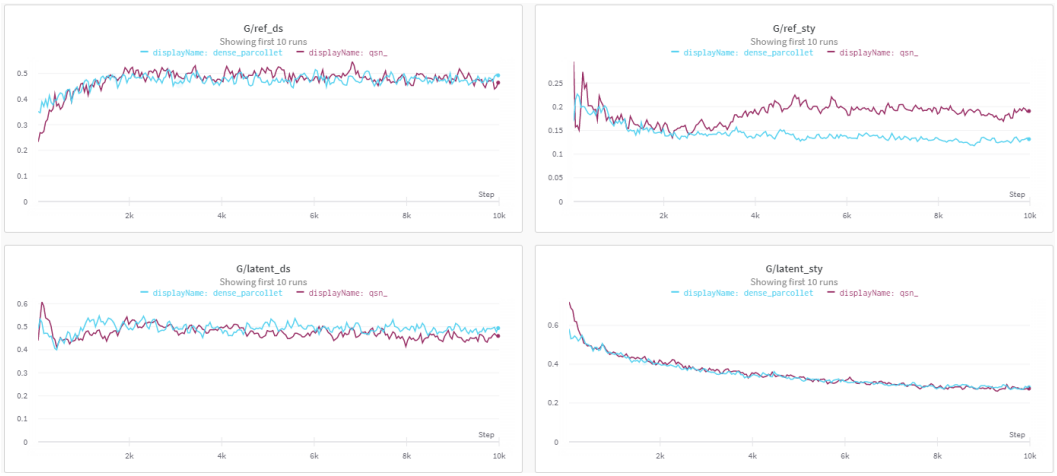


**Figure 5.5.** Generator loss functions compared, with the dense layer in cyan, without it in bordeaux


**Figure 5.6.** Generator loss functions compared, with quaternion initialization in green, with He initialization in yellow

### 5.3.4 Different $n$ for PHM approach

When using PHM we tried different values for $n$ exploiting the flexibility of this approach with hypercomplex parameterized multiplication and this change from a $n = 4$ to a $n = 3$ in a better way for the evaluation part. In particular we are solving a annoying problem in the cycle consistency loss, remind that this loss ensure

**(a)** Cycle consistency results with $n = 4$ **(b)** Cycle consistency results with $n = 3$

**Figure 5.7.** Cycle consistency loss

the fact that we are not loosing too many information of the original photo, so provides a reconstruction of the original photo after it is passed to the model once. The network in fact was unable to generate properly an image that was previously generated by it. This causes a big problem because the network provides in those cases blurred results, see 5.7 that affected the evaluation of the model in general but more in the LPIPS as showed in 5.1, we are referring to the first 10000 iterations so 10% of the training. here "REF" and "LAT" means reference guided analysis and latent guide analysis, so if we are taking the style code from the style encoder or the mapping network respectively. As said before the FID measures the feature distance between the real training images and the generated ones.

| **Method** | **FID REF** | **FID LAT** | **LPIPS REF** | **LPIPS LAT** |
|---|---|---|---|---|
| PHM with $n = 4$ | 43.6 | 47.99 | 0.155 | 0.167 |
| PHM with $n = 3$ | **40.56** | **27.77** | **0.26** | **0.339** |

**Table 5.1.** Improvement in FID results using different $n$ (10% of training)

After the experiments and the disoovery found in the previous sections, we developed the final model that performs really well and tested it against its counterpart in the real domain and also with respect to the pure quaternion approach. The improvements of the model are showed for the first 10% of the training in table 5.2.

| Method | FID REF | FID LAT |
|---|---|---|
| PHM StarGAN | 46.36 | 48.73 |
| + Quaternion Instance Normalization | 28.89 | 29.79 |
| + Last dense layer discriminator | **27.76** | 27.77 |
| + PHM with $n = 3$ | 31.61 | **20.76** |

**Table 5.2.** Improvement in FID results (10% of training)

### 5.3.5 Stargan IMT Results

I will present the results of StarGAN v2 for three different final models we have developed:

- Real StarGAN v2
- Quaternion StarGAN v2
- Parameterized Hypercomplex ($n = 3$) StarGAN v2

The number of parameters and the training time is presented in 5.3. The evaluation

| **Model** | **Params** | **Time (T)** | **Time (I)** | **Space** |
|---|---|---|---|---|
| StarGAN v2 | 87M | **421s** | 128$s$ | 306.5$MB$ |
| Quaternion StarGAN v2 | 21.8M(−75%) | 463$s$ | **127s** | **75.1MB** |
| PHM StarGAN v2 $n = 3$ | 29.2M(−67%) | 558$s$ | 136$s$ | 137$MB$ |
| PHM StarGAN v2 $n = 4$ | 21.8M(−75%) | 499$s$ | 130$s$ | 76.8$MB$ |

**Table 5.3.** Comparison of the different StarGAN v2 versions. Time of training is referred to 100 iterations, while time of Inference is to generate both latent and reference sample results.

of StarGAN v2 is divided in two major aspects, reference guided and latent guided synthesis, the first one is when we extract the style code from a reference image given by the style encoder, while the latter is when we are using the style code provided by the mapping network. I will start examining the result for the reference-guided image synthesis on CelebA-HQ.


**Figure 5.8.** Reference guided analysis comparison on CelebA-HQ dataset

The source and reference images in the first row and column are real images, the merge between them is reported in the corresponding row and column of the image 5.8. Interesting to notice that the results with n=3 results to be more guided by the reference image than the others, this justifies the results showed in table 5.4. When using the word merge before we mean that the network transforms a source image using the style of the reference image passed to the style encoder. The styles of an image could be the hairstyle or makeup or the age, in the generated photo the network preserves the pose the source image. The column images have the same identity with different styles, while the row's one share the style with different identities. The quantitative results are showed in 5.4. From those results it is possible to observe how the original approach, the real one, performs better than others in the FID, but it is not the best for what concern the LPIPS, here the PHM model outperforms the others. It's interesting to see how the different $n$ affect the parameterized hypercomplex approach, we explained this changes in the metrics results in this way, probably the blurred results when using $n = 4$ made the resulting images more similar between them and this affect the LPIPS, that in

some sense measure the diversity of the generated images. This is corrected when we switch to a smaller $n$, but we are losing on the other hand the FID, that in the previous version ($n = 4$) has more similar result to the original model.

The different results when we are changing dataset, are still something to be investigated in the future. One thing for sure that we can say about this higher results in the PHM version that performs well in the CelebA-HQ is that the two architecture are slightly different because when we switch dataset the support network, mentioned in 4.1.1.1, that helps to find edges of a face, is not present when using a different dataset, for instance animal faces. Because of the time of training we have not tested if this intuition is correct, but to do this we can run the experiment without this network that provides the heatmap of an images [37], and see if the results for the PHM StarGAN v2 changes.

| Method | CelebA-HQ | | AFHQ | |
|---|---|---|---|---|
| | FID ↓ | LPIPS ↑ | FID ↓ | LPIPS ↑ |
| StarGAN v2 | **17.16** | 0.243 | **26.07** | 0.3374 |
| Quaternion StarGAN v2 | 23.09 | 0.215 | 47.54 | 0.363 |
| PHM StarGAN v2 ($n = 4$) | 19.1 | 0.113 | 39.69 | 0.221 |
| PHM StarGAN v2 ($n = 3$) | 28.11 | **0.29** | 53.02 | **0.377** |

**Table 5.4.** Quantitative comparison on reference-guided synthesis.

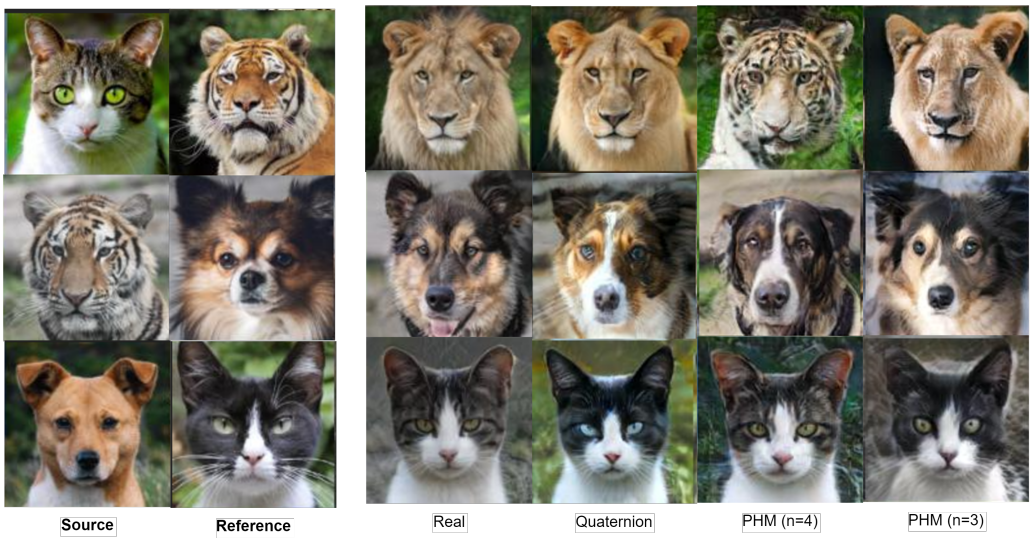


**Figure 5.9.** Reference analysis, comparison of the results for the AFHQ dataset

Interesting to notice that of the four network tested, for the first row of the

image 5.9, three of them convert a cat to a lion, while the PHM with $n = 4$ interpret the same style code to be at the end of the mix a jaguar. As mentioned before, the

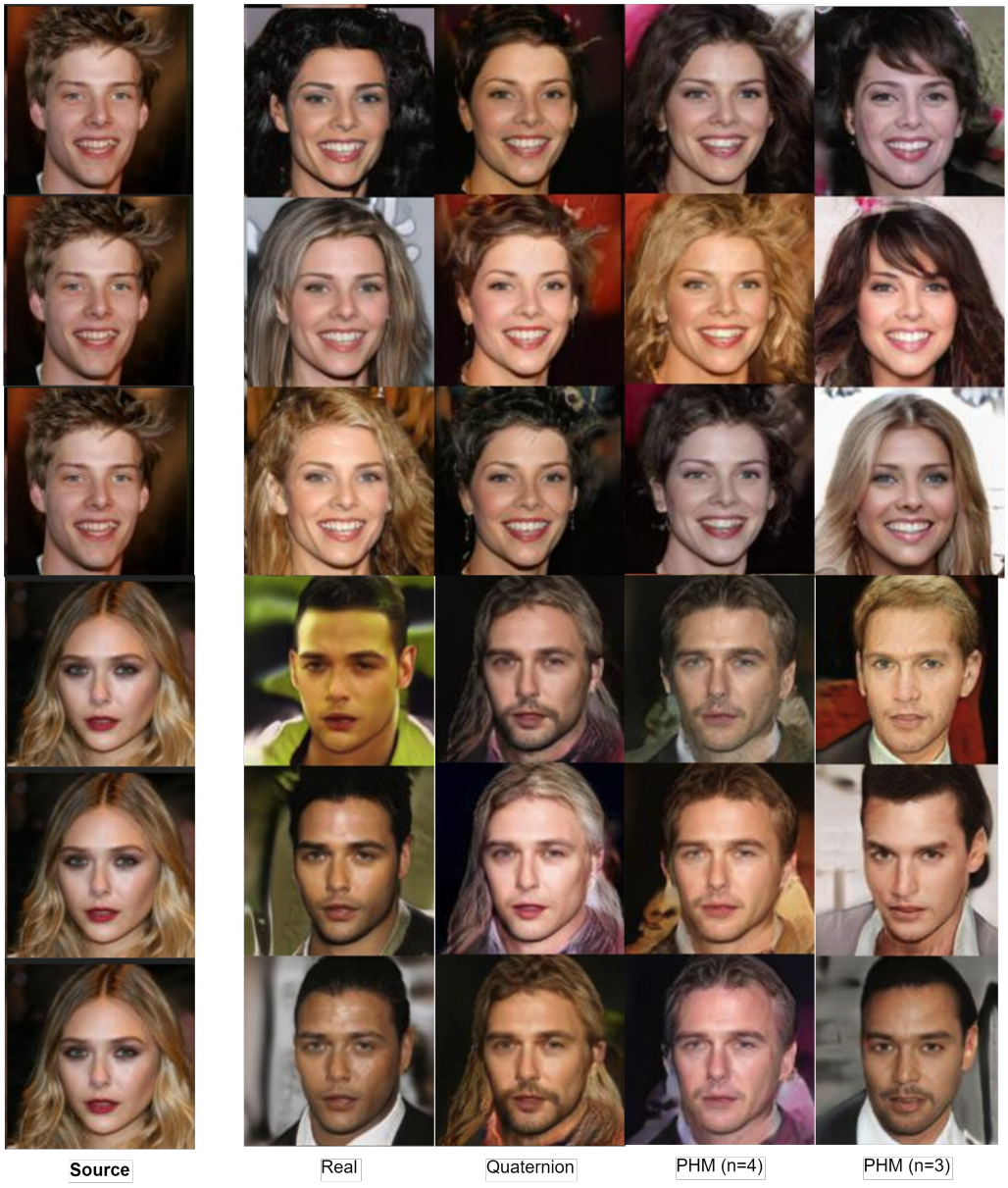


**Figure 5.10.** Latent images with $\psi = 1$. This value scales the deviation of from the average, $\psi = 1$ is equivalent to no truncation, while values towards 0 gets closer to the average, with quality improvement but a reduction in terms of variety.

other analysis that we performed on the model is the latent one, so when we are using the style code provided by the mapping network. In 5.10 the left columns there are the source images, in this analysis the target domain is randomly sampled

from latent codes. The quantitative results are showed in 5.5. From those results,


**Figure 5.11.** Latent guided analysis comparison, done on the AFHQ dataset

in the CelebA-HQ case, it is clear that the PHM model with $n = 3$ outperforms the real and the other hypercomplex approaches, nonetheless the female to male results of phm with $n = 3$ seems to be more random than the others, this justify the higher LPIPS. But again, not for what concerns the AFHQ datasets, while the LPIPS measure put an attention to the model where $n = 4$, but also in this case when we are using a canonical quaternion approach, in fact the results are really low compared to the baseline in the real domain.

We can conclude saying that our model with a number of parameters that is one third of the original one, can compare and also surpass the real approach when we are using the CelebA-HQ dataset, while on the AFHQ we saw that the real version performs better.

| Method | CelebA-HQ | | AFHQ | |
|---|---|---|---|---|
| | FID ↓ | LPIPS ↑ | FID ↓ | LPIPS ↑ |
| StarGAN v2 | 17.16 | 0.247 | **21.92** | **0.321** |
| Quaternion StarGAN v2 | 27.9 | 0.117 | 30.82 | 0.315 |
| PHM StarGAN v2 ($n = 4$) | 18.3 | 0.134 | 39.33 | 0.216 |
| PHM StarGAN v2 ($n = 3$) | **16.63** | **0.332** | 32.62 | 0.286 |

**Table 5.5.** Quantitative comparison on latent-guided synthesis

### 5.3.6 StarGAN v2 for Biomedical IMT Results

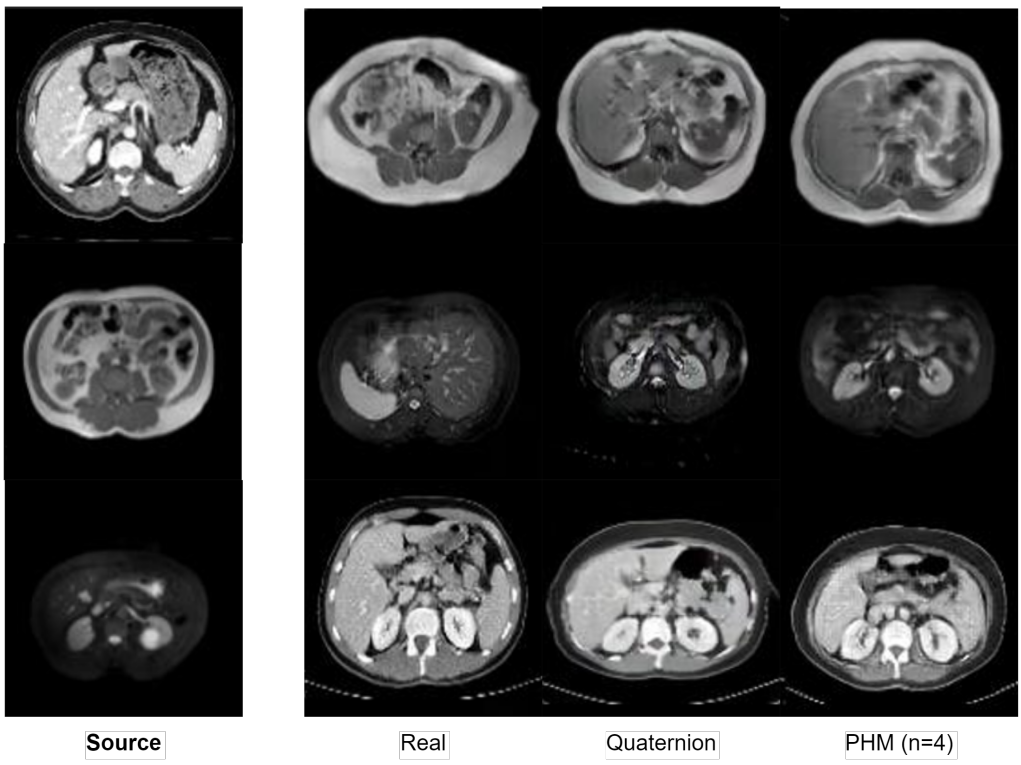


**Figure 5.12.** Latent generated images with CHAOS dataset

Since the success in the RGB domain that this network had, we want to test it also on the other dataset that is used by the TarGAN, that is a biomedical one. As anticipated in the approach section for the TarGAN in hypercomplex domain we have to deal with the different structure of the images, having this domain only one channel being gray scale images. To deal with that in this case we loaded the medical dataset using the same process for TarGAN and then saved the images, replicating the same channel to create RGB images.

The results showed are done with 50k iterations and not 100k like the other datasets because this one is really smaller wrt to the others. The quantitative results for the reference analysis are showed in 5.6, nonetheless this analysis is useless for this task of image modality translation. In fact there will be two images of different domain, for example T1 and T2, but producing a synthetic image that is not in a third domain, but in one of the two from where it started. So we will show the results of the latent analysis in 5.12. The FID value seems to be higher in both the real and the hypercomplex domain, but we are in a different world dealing with different kind of images, and we will see later on that those values are not so higher as we can think comparing to the TarGAN for example. The results are divided in CT, T1w and T2w, meaning that we are considering a mean between the target modality translation, i.e. $FID(CT) = \frac{FID(T1w \to CT) + FID(T2w \to CT)}{2}$.

| Method | FID ↓ | | | LPIPS ↑ | | |
|---|---|---|---|---|---|---|
| | CT | T1w | T2w | CT | T1w | T2w |
| StarGAN v2 | **66.3** | 61.3 | **49.33** | 0.206 | **0.184** | **0.08** |
| Quaternion StarGAN v2 | 74.6 | 72.83 | 133.6 | 0.195 | 0.147 | 0.065 |
| PHM StarGAN v2 ($n = 3$) | 78.62 | **60.84** | 62.67 | **0.239** | 0.106 | 0.020 |

**Table 5.6.** Results on Chaos dataset for reference guided synthesis

In this model we are not specializing the model to give us the target modality we want, so the generated images are still depending on some reference image as the one that provides the style code or in the latent guide the domain is sampled from a latent vector. We are still doing image to image translation but not image modality translation. The latent guided quantitative results are showed in 5.7.

| Method | FID ↓ | | | LPIPS ↑ | | |
|---|---|---|---|---|---|---|
| | CT | T1w | T2w | CT | T1w | T2w |
| StarGAN v2 | **47.75** | **51.86** | 85.59 | **0.24** | **0.2** | **0.154** |
| Quaternion StarGAN v2 | 69.16 | 66.95 | 92.96 | 0.219 | 0.176 | 0.15 |
| PHM StarGAN v2 ($n = 3$) | 67.32 | 76.89 | **74.88** | 0.056 | 0.012 | 0.024 |

**Table 5.7.** Results on Chaos dataset for latent guided synthesis

## 5.4 TarGAN experimental results

As said in the previous chapter, we will use a mixed approach to use hypercomplex networks in biomedical image modality translation. The experiments are conducted on the original model and its extension in hypercomplex domain, with a classical quaternion approach and a PHM approach. The tasks that accomplish this model are two, translation and segmentation, so we will evaluate them differently and separately. A comparison of the models in terms of parameters and time of training is showed in 5.8

| Model | Params | Time (T) | Time (I) | Space |
|---|---|---|---|---|
| TarGAN | 125.2M | **315s** | $49s$ | $477MB$ |
| Quaternion TarGAN | 31.3M($-75\%$) | $588s$ | $58s$ | $119MB$ |
| Quaternion-mixed TarGAN | 40.2M($-67\%$) | $720s$ | **32s** | $153MB$ |
| PHM TarGAN $n = 4$ | 31.3M($-75\%$) | $800s$ | $69s$ | $119MB$ |
| PHM-mixed TarGAN $n = 4$ | 40.2M($-67\%$) | $949s$ | $37s$ | $153MB$ |

**Table 5.8.** Comparison of parameters and time of training for 1 epoch and for inference the sampling time of the three different domains, using an NVIDIA P100

### 5.4.1 TarGAN for Biomedical IMT

To evaluate the task of image modality translation, we will use both a standard FID, and a FID used in the original work that is a composition of images (16 slices) passed into a different model and then evaluated with the Frechet distance, so to compare them fairly with the StarGAN v2 we decided to use both the methods.

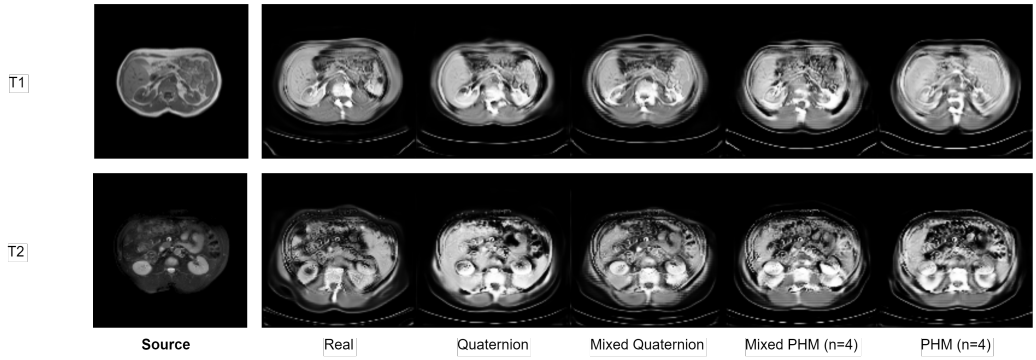


**Figure 5.13.** TarGAN IMT for $T1 \rightarrow CT$ and $T2 \rightarrow CT$

The quantitative results for the classical FID are showed in 5.9. The results

in terms of what model performs better are not clearly visible, we have for sure the quaternion classical approach that surpasses the PHM approach, but when it comes to compare the original model to its quaternion version, we have different results, but for two of the three modalities (CT, T2w) the quaternion surpass its counterpart. It's strange that the PHM model that until now seems to be the best solution for the model seen so far, performs so bad, but we think that the depthwise concatenation is not helping to explore concretely the intrachannel relations, like we have seen in the RGB approach with the StarGAN v2.
So we can conclude comparing also to the qualitative results:

- In 5.13 we see the translation to the CT domain from T1 weighted and T2 weighted MRI
- In 5.14 we see the translation to the T1 domain from CT and T2 weighted MRI
- In 5.15 we see the translation to the T2 domain from CT and T1 weighted MRI

All of the translation are reported with the different models to do a comparison, this comparison, differently from other datasets, is hard to evaluate if we are not doctors, but seeing the quantitative results in 5.10 we can conclude that the PHM model can be easily interchanged with its counterpart saving 1/4 of the parameters without losing quality, and also enhancing the results in some cases.

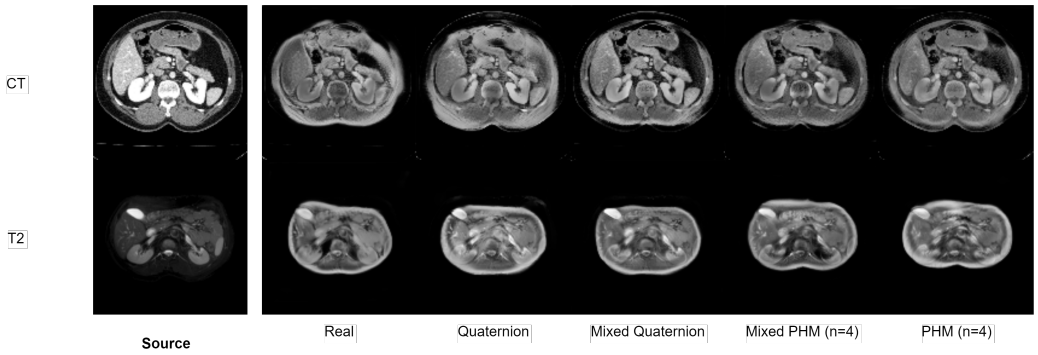


**Figure 5.14.** TarGAN IMT for $CT \rightarrow T1$ and $T2 \rightarrow T1$

| Method | FID ↓ | | |
|---|---|---|---|
| | CT | T1w | T2w |
| TarGAN | 163.77 | **119.79** | 145.73 |
| Quaternion TarGAN | **157.11** | 131.29 | **128.17** |
| Quaternion-mixed TarGAN | 201.24 | 144.94 | 167.23 |
| PHM TarGAN | 193.02 | 143.08 | 165.98 |
| PHM-mixed TarGAN | 181.92 | 130.78 | 164.53 |

**Table 5.9.** FID results for different implementations of TarGAN

The FID results compared with the StarGAN v2, are really poor, we think that this is normal because we are doing two tasks at the same time in the TarGAN, this can also explain why the paper concentrates in the usage of a different FID calculus, the original explanation is because we want to consider only zone that are really interest for the biomedical case.

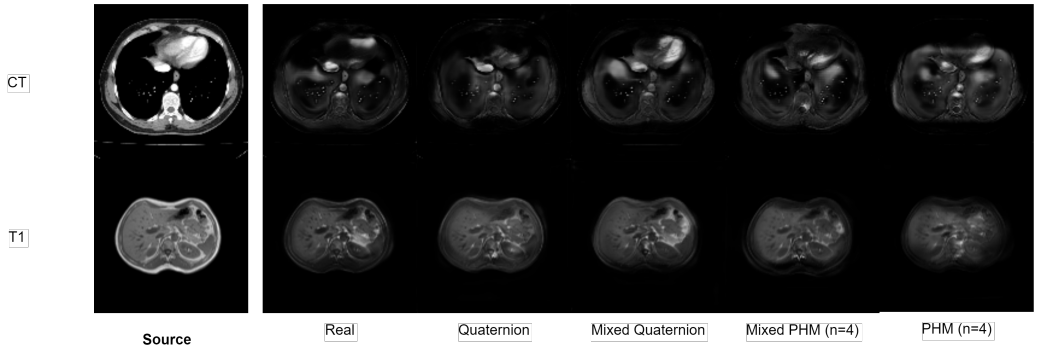


**Figure 5.15.** TarGAN IMT for $CT \rightarrow T2$ and $T1 \rightarrow T2$

The original work use a network as support to use evaluate only meaningful features. From the last layer of the encoder of the pre-trained Models Genesis [46] the feature vectors are extracted and processed with a global average pooling layer. Models Genesis is trained with 3D medical volume data, but our dataset contains only 2D images, so the idea of concatenating every 16 axial slices forming a 3D volume patch, the results are showed in table 5.10. Those results are in some sense turning the table because TarGAN with PHM layers has better performances when it comes to extract those features important for medical imaging.

| Method | FID ↓ | | |
|---|---|---|---|
| | CT | T1w | T2w |
| TarGAN | 0.0931 | 0.0971 | 0.0979 |
| Quaternion TarGAN | **0.0907** | 0.0986 | 0.0953 |
| Quaternion-mixed TarGAN | 0.0918 | 0.0877 | **0.0901** |
| PHM TarGAN | 0.0913 | 0.0909 | 0.0943 |
| PHM-mixed TarGAN | 0.0939 | **0.0832** | 0.1041 |

**Table 5.10.** FID results, computed with [46], for different implementations of TarGAN

### 5.4.2 TarGAN Biomedical Segmentation

As anticipated before this network does also a task of segmentation of the translated image, the following evaluation are performed considering the 2D image generated by the network, considering the fact that dataset is 2D. The quantitative results are showed in 5.11. Following the image to image translation results, the quaternion classical approach outperform for the majority of the metrics, but in this task the PHM redeems itself having good results in the segmentation, in most of the cases it goes better than the original. To compare the segmentation done by the network

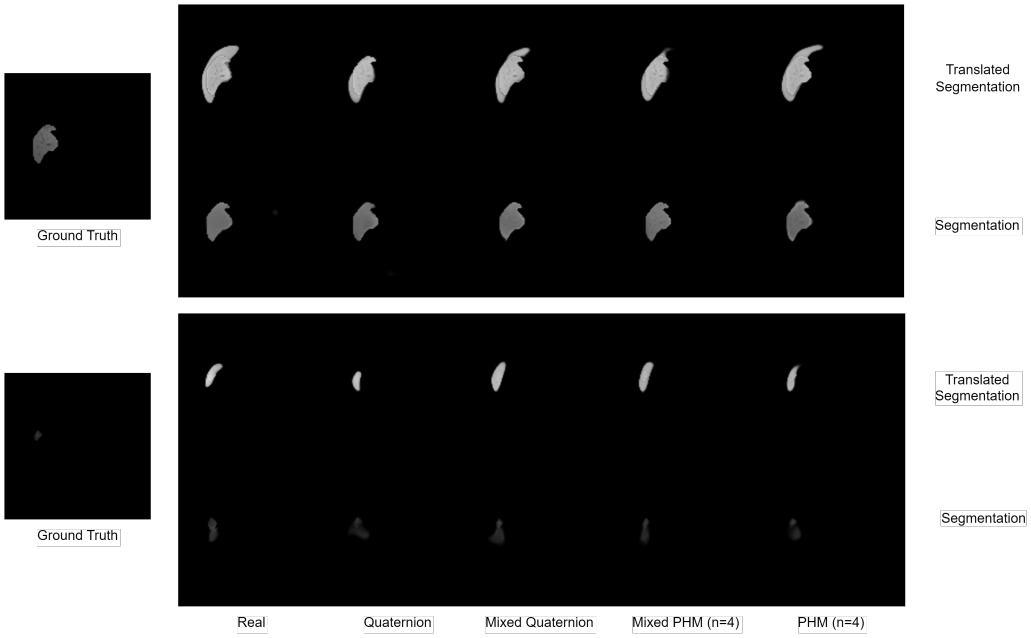


**Figure 5.16.** Segmentation to CT domain

with the one provided as ground truth we will show both the segmentend images. For instance in 5.16 we could see the ground truth in its original domain and the

correspondent segmentation in the same domain, and in the translated domain, in this case the CT.

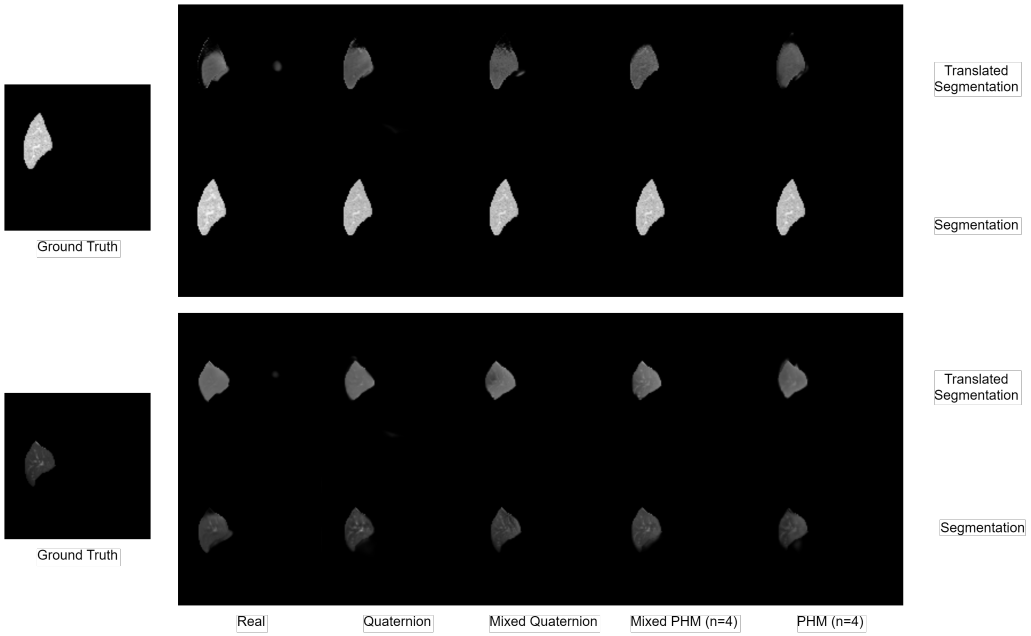


**Figure 5.17.** Segmentation to T1 domain

| Method | DICE (%) ↑ | | | RAVD (%) ↓ | | | S-score (%) ↑ | | |
|---|---|---|---|---|---|---|---|---|---|
| | CT | T1w | T2w | CT | T1w | T2w | CT | T1w | T2w |
| TarGAN | 91.96 | 87.83 | 85.68 | 17.01 | 5.25 | 10.44 | 70.16 | 73.91 | 71.36 |
| Quaternion TarGAN | **93.58** | 89.53 | 88.79 | **12.61** | 7.53 | 16.8 | **80.17** | **85.33** | **85.49** |
| Quaternion-mixed TarGAN | 92.74 | 91.82 | 90.16 | 15.38 | 0.16 | 11.19 | 79.28 | 83.38 | 81.35 |
| PHM TarGAN | 91.94 | 87.72 | 89.04 | 16.55 | 2.7 | 8.65 | 75.12 | 79.83 | 78.97 |
| PHM-mixed TarGAN | 91.43 | **93.56** | **91.99** | 17.91 | **1.87** | **8.39** | 75.61 | 78.63 | 78.39 |

**Table 5.11.** Segmentation scores for TarGAN

The qualitative results for the other domains are showed in 5.17 and 5.18.

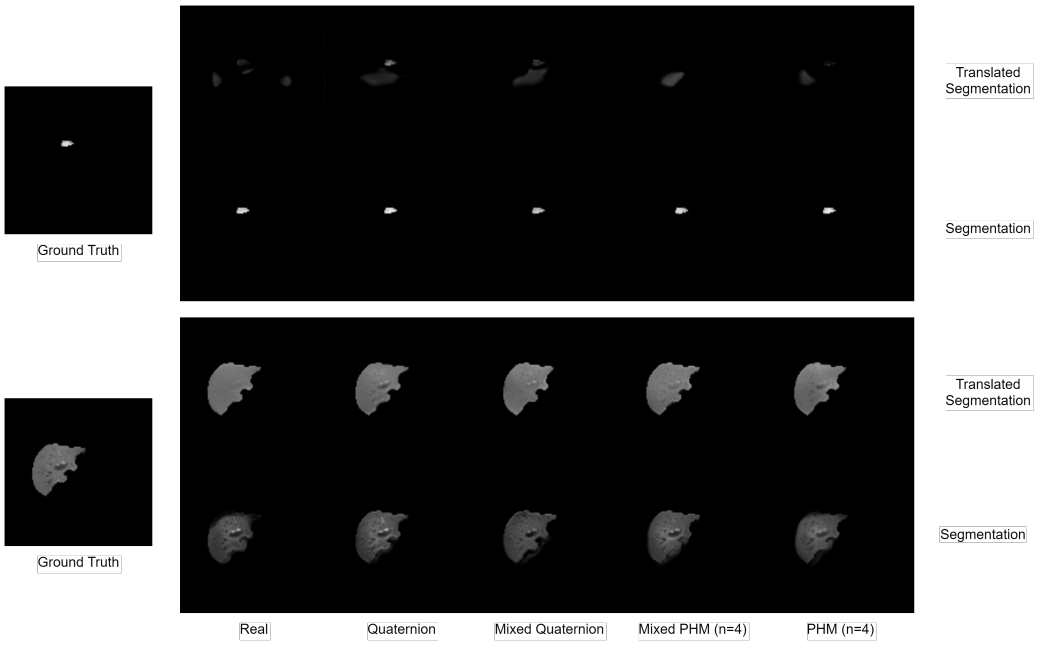


**Figure 5.18.** Segmentation to T2 domain

Also in this task, we can conclude that the hypercomplex models are performing better than the original approach.

# Chapter 6

# Conclusions

In the previous chapter we presented the results of my studies on hypercomplex neural networks specialized in the generative deep learning. The results obtained are surprisingly good in some cases, like the StarGAN v2 using PHM layers, and also interestingly when it comes to the classical quaternion in the TarGAN. There are different future works for this thesis.

In general the newest hypercomplex layers showed how they can subsume the Hamilton product and being more flexible, this permits to explore different possibilities for different tasks. Being more specific, we have used different $n$ for the StarGAN v2 resulting in different results, some better and some worst wrt to the real counterpart (the original work), depending also on the dataset we are analysing, this can be further investigated to find a connection between those differences.

For what concerns the TarGAN, we have tested the model using a well known solution to handle 1 channels images, but this solution, coming from the fact that we are in a hypercomplex domain where we try to exploit the intra channels information, it is not the best. In recent works, there is a new approach that tries to solve the problem using different channels to emulate a RGB image, so not just replicating the missing channels, this can be investigated in our hypercomplex approach to provides a more sophisticated solution and to improve maybe our current results. The results for this type of models requires ton of time to train and also every training requires time to find the correct hyperparameters, this is one of the main issues that affect in part the work presented.

I was limited by the technology, in fact we used for most of the time NVIDIA GPUs provided by my Google Colab subscription (P100 or Tesla T4), but there are restriction in the time of usage. Because of that we narrowed the experiment in some sense because we used a 128x128 image size for all the training instead of a 256x256 as the original work. This can affect the metrics, we see for example an improvement in the LPIPS when using the quaternion approach with the 256 with respect to the 128, but because of those limitations (the 256x256 does not fit in the V-Ram memory of the GPU we used) we preferred to stick to a smaller size.
A future work could be experimenting using the original sizes, seeing that there are improvement in the metrics this could lead to a even better model. I also used whenever possible the more powerful NVIDIA V100 provided by the ISPAMM laboratory.
More or less, even with the newest graphics card, a complete experiment takes about six days to complete and to evaluate. A part from that we really enjoyed and I am even more curios to see what we can do in the future with this approach for GAN to do Image modality translation for biomedical task or a generic Image to Image Translation.

# Bibliography

[1] Aumentado-Armstrong, T., Levinshtein, A., Tsogkas, S., Derpanis, K. G., and Jepson, A. D. Cycle-consistent generative rendering for 2d-3d modality translation. In *2020 International Conference on 3D Vision (3DV)*, pp. 230–240 (2020). doi:10.1109/3DV50981.2020.00033.

[2] Chen, J., Wei, J., and Li, R. Targan: Target-aware generative adversarial networks for multi-modality medical image translation (2021). arXiv:2105.08993.

[3] Choi, Y., Choi, M., Kim, M., Ha, J.-W., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8789–8797 (2018). doi:10.1109/CVPR.2018.00916.

[4] Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. Stargan v2: Diverse image synthesis for multiple domains (2020). arXiv:1912.01865.

[5] Dar, S. U., Yurt, M., Karacan, L., Erdem, A., Erdem, E., and Cukur, T. Image synthesis in multi-contrast MRI with conditional generative adversarial networks. *IEEE Transactions on Medical Imaging*, **38** (2019), 2375. Available from: https://doi.org/10.1109/tmi.2019.2901750, doi:10.1109/tmi.2019.2901750.

[6] Dice, L. R. Measures of the amount of ecologic association between species. *Ecology*, **26** (1945), 297. Available from: http://www.jstor.org/stable/1932409.

[7] Dumoulin, V. and Visin, F. A guide to convolution arithmetic for deep learning. *ArXiv e-prints*, (2016). arXiv:1603.07285.

[8] Gao, L., Hong, D., Yao, J., Zhang, B., Gamba, P., and Chanussot, J. Spectral superresolution of multispectral imagery with joint sparse and low-rank learning. *IEEE Transactions on Geoscience and Remote Sensing*, **59** (2021), 2269. doi:10.1109/TGRS.2020.3000684.

[9] Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (edited by Y. W. Teh and M. Titterington), vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256. PMLR, Chia Laguna Resort, Sardinia, Italy (2010). Available from: https://proceedings.mlr.press/v9/glorot10a.html.

[10] Goodfellow, I. Nips 2016 tutorial: Generative adversarial networks (2017). arXiv:1701.00160.

[11] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks (2014). arXiv:1406.2661.

[12] Grassucci, E., Cicero, E., and Comminiello, D. Quaternion generative adversarial networks (2021). arXiv:2104.09630.

[13] Grassucci, E., Zhang, A., and Comminiello, D. Lightweight convolutional neural networks by hypercomplex parameterization (2021). arXiv:2110.04176.

[14] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015). arXiv:1502.01852.

[15] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems* (edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach,

R. Fergus, S. Vishwanathan, and R. Garnett), vol. 30. Curran Associates, Inc. (2017). Available from: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf.

[16] Hiasa, Y., Otake, Y., Takao, M., Matsuoka, T., Takashima, K., Carass, A., Prince, J. L., Sugano, N., and Sato, Y. Cross-modality image synthesis from unpaired data using cyclegan. In *Simulation and Synthesis in Medical Imaging* (edited by A. Gooya, O. Goksel, I. Oguz, and N. Burgos), pp. 31–41. Springer International Publishing, Cham (2018).

[17] Hiasa, Y., Otake, Y., Takao, M., Matsuoka, T., Takashima, K., Prince, J. L., Sugano, N., and Sato, Y. Cross-modality image synthesis from unpaired data using cyclegan: Effects of gradient consistency loss and training data size (2018). arXiv:1803.06629.

[18] Huang, X. and Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 1510–1519 (2017). doi:10.1109/ICCV.2017.167.

[19] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks (2018). arXiv:1611.07004.

[20] Karras, T., Aila, T., Laine, S., and Lehtinen, J. Progressive growing of gans for improved quality, stability, and variation (2018). arXiv:1710.10196.

[21] Kavur, A. E., Selver, M. A., Dicle, O., Barış, M., and Gezer, N. S. CHAOS - Combined (CT-MR) Healthy Abdominal Organ Segmentation Challenge Data (2019). Available from: https://doi.org/10.5281/zenodo.3362844, doi:10.5281/zenodo.3362844.

[22] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger), vol. 25. Curran Associates, Inc. (2012). Available from: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[23] Li, J., Ling, Z., Niu, L., and Zhang, L. Bi-directional domain translation for zero-shot sketch-based image retrieval. *ArXiv*, **abs/1911.13251** (2019).

[24] Mao, Q., Lee, H.-Y., Tseng, H.-Y., Ma, S., and Yang, M.-H. Mode seeking generative adversarial networks for diverse image synthesis (2019). arXiv:1903.05628.

[25] Mescheder, L., Geiger, A., and Nowozin, S. Which training methods for gans do actually converge? (2018). arXiv:1801.04406.

[26] Mirza, M. and Osindero, S. Conditional generative adversarial nets (2014). arXiv:1411.1784.

[27] Olut, S., Sahin, Y. H., Demir, U., and Unal, G. Generative adversarial training for mra image synthesis using multi-contrast mri (2018). arXiv:1804.04366.

[28] Parcollet, T., Morchid, M., and Linarès, G. A survey of quaternion neural networks. *Artificial Intelligence Review*, **53** (2019), 2957.

[29] Parcollet, T., Morchid, M., and Linarès, G. Quaternion convolutional neural networks for heterogeneous image processing. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8514–8518 (2019). doi:10.1109/ICASSP.2019.8682495.

[30] Parcollet, T., Zhang, Y., Morchid, M., Trabelsi, C., Linarès, G., Mori, R. D., and Bengio, Y. Quaternion convolutional neural networks for end-to-end automatic speech recognition (2018). arXiv:1806.07789.

[31] Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., and Efros, A. A. Context encoders: Feature learning by inpainting (2016). arXiv:1604.07379.

[32] Salimi, A., Pourmina, M. A., and Moin, M.-S. Fully automatic prostate segmentation in MR images using a new hybrid active contour-based approach. *Signal, Image and Video Processing*, **12** (2018), 1629. Available from: https://doi.org/10.1007/s11760-018-1320-y, doi:10.1007/s11760-018-1320-y.

[33] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision (2015). arXiv: 1512.00567.

[34] Ujang, B. C., Took, C. C., and Mandic, D. P. Quaternion-valued nonlinear adaptive filtering. *IEEE Transactions on Neural Networks*, **22** (2011), 1193. doi:10.1109/TNN.2011.2157358.

[35] Ulyanov, D., Vedaldi, A., and Lempitsky, V. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4105–4113 (2017). doi:10.1109/CVPR.2017.437.

[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need (2017). arXiv:1706.03762.

[37] Wang, X., Bo, L., and Fuxin, L. Adaptive wing loss for robust face alignment via heatmap regression (2020). arXiv:1904.07399.

[38] Wu, Y. and He, K. Group normalization (2018). arXiv:1803.08494.

[39] Xin, B., Hu, Y., Zheng, Y., and Liao, H. Multi-modality generative adversarial networks with tumor consistency loss for brain mr image synthesis (2020). arXiv:2005.00925.

[40] Yang, Q., Li, N., Zhao, Z., Fan, X., Chang, E. I.-C., and Xu, Y. MRI cross-modality image-to-image translation. *Scientific Reports*, **10** (2020). Available from: https://doi.org/10.1038/s41598-020-60520-6, doi:10.1038/s41598-020-60520-6.

[41] Yu, B., Zhou, L., Wang, L., Shi, Y., Fripp, J., and Bourgeat, P. Ea-GANs: Edge-aware generative adversarial networks for cross-modality MR image synthesis. *IEEE Transactions on Medical Imaging*, **38** (2019), 1750. Available from: https://doi.org/10.1109/tmi.2019.2895894, doi: 10.1109/tmi.2019.2895894.

[42] Zhang, A., Tay, Y., Zhang, S., Chan, A., Luu, A. T., Hui, S. C., and Fu, J. Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters (2021). arXiv:2102.08597.

[43] Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. The unreasonable effectiveness of deep features as a perceptual metric (2018). arXiv:1801.03924.

[44] Zhang, Z., Yang, L., and Zheng, Y. Translating and segmenting multimodal medical volumes with cycle- and shape-consistency generative adversarial network (2019). arXiv:1802.09655.

[45] Zhao, H., Li, H., Maurer-Stroh, S., and Cheng, L. Synthesizing retinal and neuronal images with generative adversarial nets. *Medical Image Analysis*, **49** (2018), 14. Available from: https://doi.org/10.1016/j.media.2018.07.001, doi:10.1016/j.media.2018.07.001.

[46] Zhou, Z., Sodha, V., Rahman Siddiquee, M. M., Feng, R., Tajbakhsh, N., Gotway, M. B., and Liang, J. Models genesis: Generic autodidactic models for 3d medical image analysis. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019* (edited by D. Shen, T. Liu, T. M. Peters, L. H. Staib, C. Essert, S. Zhou, P.-T. Yap, and A. Khan), pp. 384–393. Springer International Publishing, Cham (2019).

[47] Zhu, D., Liu, S., Jiang, W., Gao, C., Wu, T., Wang, Q., and Guo, G. Ugan: Untraceable gan for multi-domain face translation (2019). arXiv:1907.11418.

[48] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251 (2017). doi:10.1109/ICCV.2017.244.

[49] Zhu, J.-Y., Zhang, R., Pathak, D., Darrell, T., Efros, A. A., Wang, O., and Shechtman, E. Toward multimodal image-to-image translation (2018). arXiv:1711.11586.