## Neural Network 2020

# Project Presentation

Francesco Starna Gianmarco Bracalello

*Sapienza University of Rome*

SAPIENZA
UNIVERSITÀ DI ROMA

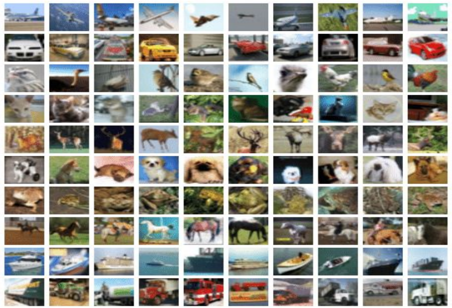
# Deep Complex Network

## Project

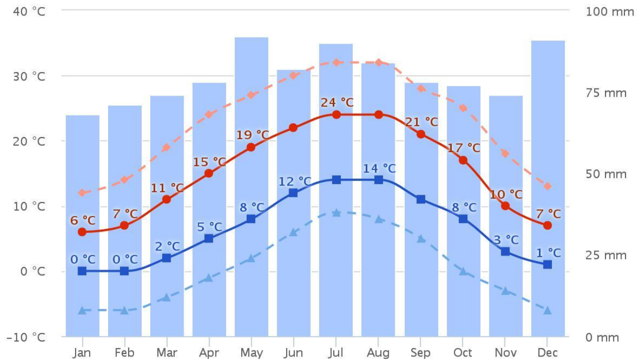
# Project Goal

Compare real residual network and complex residual network on following tasks:

- **Image Recognition**
  - CIFAR 10, CIFAR 100



- **Time Series Forecasting**
  - Weather Dataset



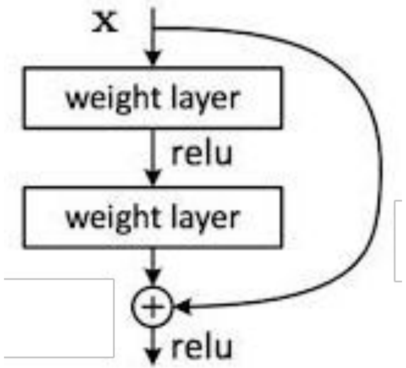

Using Tensorflow all computation are done in real context.

# Complex Residual Network

ResNet exploits advantages of deep network using shortcut paths added to output of each blocks.

Complex valued layer:

- Complex Weight Initialization
- Complex Batch Normalization
- Complex Convolution
- Imaginary Learning Block
- Complex Dropout
- Complex Activation Functions



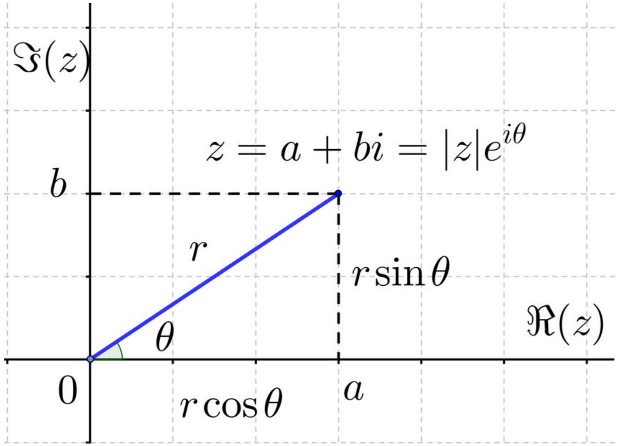
# Complex Weight Initialization

Using the polar representation of complex numbers initialize weights in a proper way

$$W = |W|e^{i\theta} = \Re(W) + i\Im(W)$$

|W| is computable from Rayleigh distribution

θ varies between -π and π




θ is the phase and |z| is the magnitude

# Whitening 2D vectors

In order to optimize the model training and performance we scale the data by the inverse square root of the covariance matrix V. Different from classic real-valued batch normalization.

$$\tilde{x} = (V)^{-\frac{1}{2}}(x - \mathbb{E}[x]) \qquad V = \begin{bmatrix} Cov(\Re(x), \Re(x)) & Cov(\Re(x), \Im(x)) \\ Cov(\Im(x), \Re(x)) & Cov(\Im(x), \Im(x)) \end{bmatrix}$$

It ensures:

- Zero mean $\mu = 0$
- Covariance one $\Gamma = 1$
- Zero correlation $C = 0$

# Complex Batch Normalization

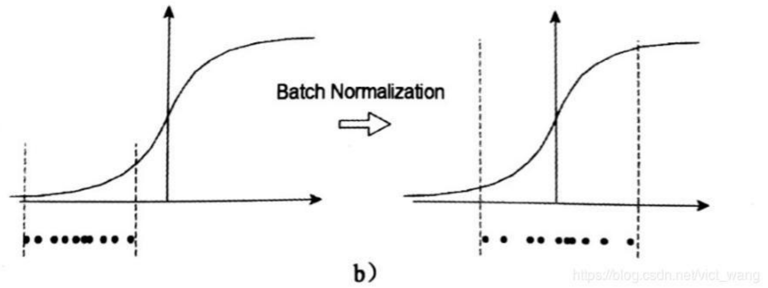
Same as the real valued batch normalization two learnable parameters are used:

- a scaling parameter

$$\gamma = \begin{pmatrix} \gamma_{rr} & \gamma_{ri} \\ \gamma_{ri} & \gamma_{ii} \end{pmatrix}$$

- a shift parameter

$$\beta = 0$$



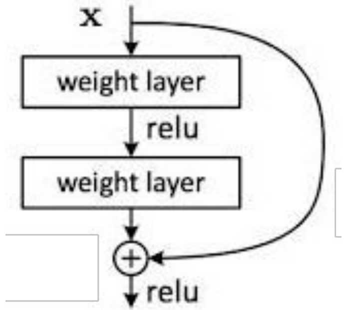

The batch normalization becomes:

$$BN(\tilde{x}) = \gamma \tilde{x} + \beta$$

# Layer Customization

**Imaginary Learning Block**: 2 real residual blocks

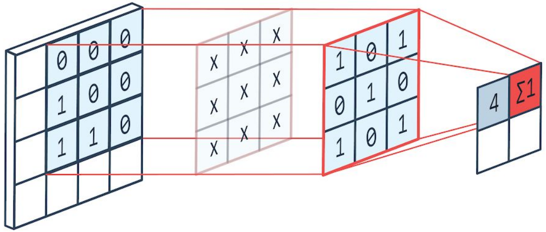
Real Input



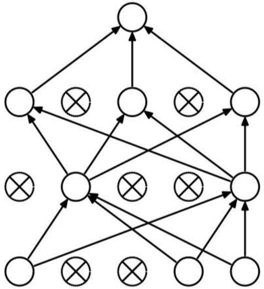

Complex Output

**Complex Convolution**: real and imaginary kernels

$$W * h = (A * x - B * y) + i(B * x + A * y)$$



**Complex Dropout**





SAPIENZA

# Complex Activation Functions

**ModReLU**: activation depending on magnitude, phase and a learnable parameter

$$modReLU(z) = ReLU(|z| + b)e^{i\theta_z} = \begin{cases} (|z| + b)\frac{z}{|z|} & |z| + b \geq 0 \\ 0 & otherwise \end{cases}$$

**CReLU**: applies ReLU on real and imaginary part of the input computing the resulting complex number

$$\mathbb{C}ReLU(z) = ReLU(\Re(z)) + iReLU(\Im(z))$$

**ZRelu**: is simply the input if the phase is in [0, $\pi/2$] and 0 otherwise

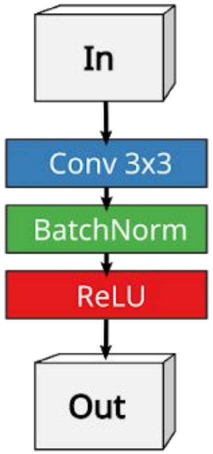
$$zReLU = \begin{cases} z & \theta_z \in [0, \pi/2] \\ 0 & otherwise \end{cases}$$

# Architecture

**Pre-Residual Block layers:**
CConv ⟶ CBatch Normalization ⟶ CActivation

**3 Stages Residual Blocks:**
CBatch Normalization ⟶ CActivation ⟶ CConv ⟶ CDropout

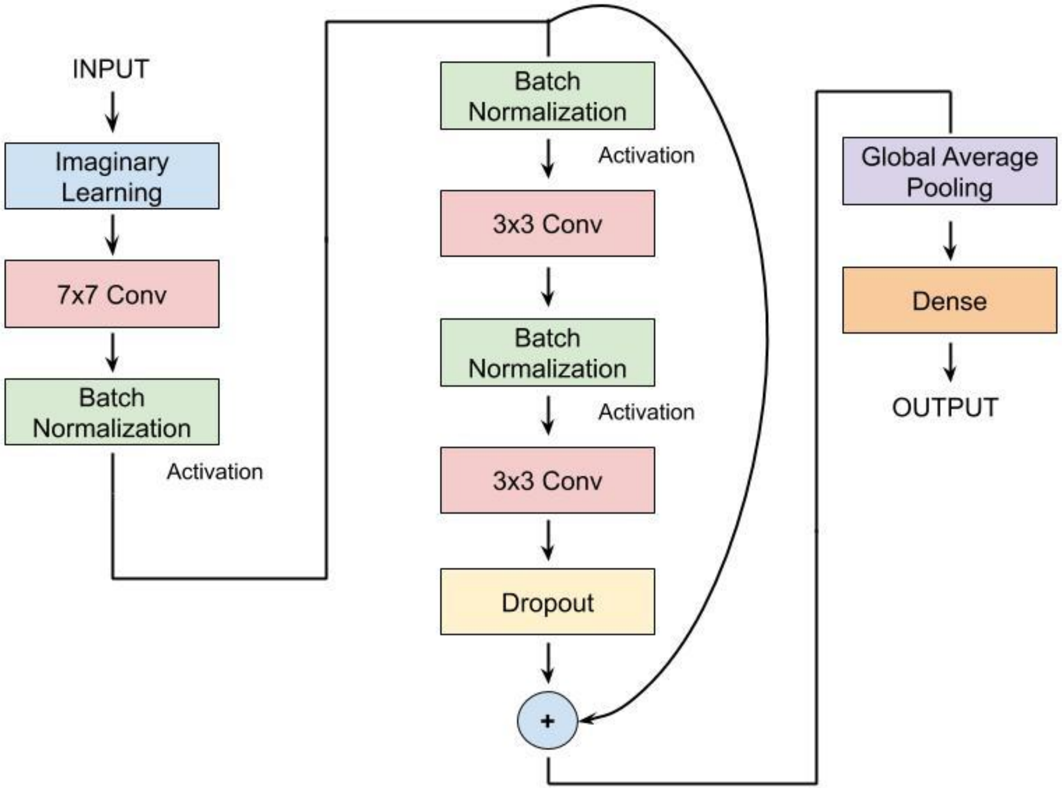
**post-Residual Block layers:**
GlobalAveragePooling ⟶ FullyConnectedLayer




To guarantee a comprehensive comparison between real and complex networks three different architectures have been used:

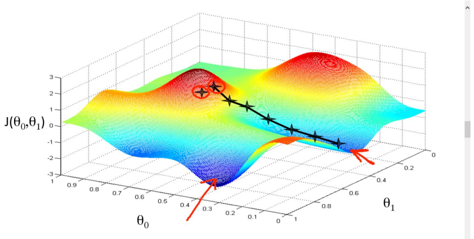
- Wide and Shallow (WS) Blocks = 14 F = [12, 24, 48]
- In Between (IB) Blocks = 17 F = [11, 22, 44]
- Deep and Narrow (DN) Blocks = 20 F = [10, 20, 40]

# Neural Network



# Training

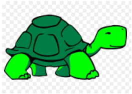
**Optimizers**: SGD (Robbins, 2007), Adam (Kingma and Ba, 2017)




## Image Recognition

CIFAR10-100

- SGD
- Nesterov momentum 0.9
- gradient clipping 1.0
- learning rate scheduler
- 200 epochs

slower 

## Time Series Forecasting

1h, 24h windowing

- Adam
- gradient clipping 1.0
- learning rate 0.001
- 120 epochs

faster 



SAPIENZA

# Results

## Image Recognition

## Time Series Forecasting

| Architecture | Blocks | Filters | Activation | Image Recognition | | Time Series | |
|---|---|---|---|---|---|---|---|
| | | | | CIFAR10 | CIFAR100 | 1h | 24h |
| **Wide-Shallow** | 14 | [12,24,48] | ModRelu | 0.767 | 0.524 | 0.902 | 0.892 |
| | | | CRelu | 0,801 | 0,613 | 0.934 | **0.946** |
| | | | ZRelu | 0,788 | - | 0,937 | 0.921 |
| Real Valued | 16 | [18,36,72] | Relu | **0.873** | **0,709** | **0.940** | 0.930 |
| **In-Between** | 17 | [11,22,44] | ModRelu | 0,740 | 0.503 | 0,919 | 0,925 |
| | | | CRelu | 0,783 | 0.605 | 0,935 | **0,941** |
| | | | ZRelu | 0,758 | - | 0,920 | 0,926 |
| Real Valued | 19 | [16,32,64] | Relu | **0,851** | **0,617** | **0,936** | 0,936 |
| **Deep-Narrow** | 20 | [10,20,40] | ModRelu | 0,703 | 0,510 | 0,905 | 0,941 |
| | | | CRelu | 0,794 | 0,626 | **0,931** | 0,937 |
| | | | ZRelu | 0,771 | - | 0,928 | 0,928 |
| Real Valued | 23 | [14,28,56] | Relu | **0,866** | **0,678** | 0,899 | **0,949** |

# Thanks