IE663: Advanced Topics in Deep Learning **Jan-May 2021**

Project Report : Deep Isometric Learning for Visual Recognition

Team Name: Team Akatsuki *Team Members: 18i190006, 19i190008*

Abstract

Though training deep convolutional networks is inherently difficult, but designing architecture for them is also not an easy task. Initialisation, normalization, residual connections, etc. are indispensable components of modern ConvNets, one can not imagine to train them without these components and achieve good performance. This project work aims to show that deep ConvNets can achieve good performance without using normalization nor skip connections. This can be achieved with the help of isometric learning where we enforce the convolution kernels to be near isometric during initialization and training, we also use SReLU non-linear activation to maintain isometry in the network. In addition to the ISONet and R-ISONet that were proposed in the original work, we created ISO-UNet which can be used for semantic image segmentation task. We have done experiments on image classification, object detection and image segmentation task and found that the isometric networks can achieve performance on par with (or even better in some case) the standard networks like ResNet and UNet.

1 Introduction

Deep Learning techniques have achieved the state of the art result in a wide range of visual recognition tasks using convolutional networks (ConvNets) and its variants. Before 2010, training ConvNets (even for 10 layers) was a difficult task due to the lack of resources and good architectural components. After 2010, many researches were carried out where a wide variety of network architectural components including novel nonlinear activation, weight initialization, weight regularization, and residual learning techniques were proposed to help ConvNets train even for more than 100 layers efficiently.

Now, the architecture design for ConvNets becomes a difficult task due to abundance of existing architectural components and the diversity of their design principles. The natural challenge here arise is that which combination of components should we use to construct our network(s)? This challenge motivates us to ask the following question: *Is there a central guiding principle for training very deep ConvNets?*[10]

According to Qi et al. ([10]), the answer to previous question is **isometric learning**. The convolutional networks created using isometric learning technique (calling them ISONets) tends to preserve the inner product (details in section 3) for both backward and forward propagation at every layer. In particular, when creating ISONets, we enforce the convolution kernels to be identity at initialization while enforcing them near orthogonal during training, and the Shifted ReLU (SReLU) non-linear activation was used to maintain network isometry. We can obtain SReLU by mofifying ReLU, i.e. by shifting regular ReLU towards being an identity around the origin with a learnable parameter. Mathematically, $\text{SReLU}_b(y) = \max(y, b)$, where b is a learnable parameter and y is the input.

Our contributions to this project are mentioned in section 1.1. We provide a survey of existing literature in Section 2. Our proposal for the project is described in Section 3. We give details on data set in Section 4, and on experiments in Section 5. Section 6 consists of a summary of results obtained with our experiments. A description of future work is given in Section 7. We conclude with a short summary and pointers to forthcoming work in Section 8.

1.1 Contributions:

Our project makes following contribution:

- Semantic image segmentation is an important computer vision task. One requires to train a UNet[11] (or its variant) with batch normalization layers, to get state of the art performance on segmentation task. In this project, we propose ISO-UNet which **does not require any batch normalization layer** to give excellent performance. In UNet, one can face the problem of kernel initialization, like what values to give to kernels initially in order to achieve good performance, however, ISO-UNet does not suffer any such problem. We performed experiment on brain MRI data set and found that ISO-UNet is performing on par with UNet.
- In original work by Qi et al. [10], they only mentioned that accuracy on object detection task for COCO data set is higher as compared to ResNet, there was no code available to confirm the claim. Therefore, we also code and built R-ISONet + Faster RCNN framework for object detection and perform experiment on COCO data set.

2 Literature Survey

2.1 In Context of Isometric Learning

Isometry is not a new concept, but it was only after 2010 that deep learning community adopted it. For improving existing network training and performance, numerous instantiations of isometric principle have been implicitly or explicitly suggested, exploited and studied in the literature, often as regularization or an additional heuristic. The notion of isometry was first explored in the context of weight initialization and then in diverse contexts such as weight regularization, design of non-linear activation, and training techniques for residual networks [10].

Weight initialization and regularization plays an important role in isometric learning, therefore it is important to discuss a few related work in the context of weight initialization. Early works (around 2010) on weight initialization are based on the principle that the variance of the signal maintains a constant level as it propagates forward or backward through multiple layers ([6]; [3]). A popular method that provides such a guarantee is the Kaiming Initialization ([4]) which derives a proper scaled Gaussian initialization for weights in vanilla convolutional networks with ReLU activation functions. Derivations for general activation functions beyond ReLU is more difficult, but nonetheless can be achieved by working in the regime where the network is infinitely wide using mean-field theory ([9]). From the perspective of isometry, the aforementioned works guarantee that the average of the squared singular values of the input-output Jacobian matrix is close to 1. However, this does not mean that the Jacobian is an isometry, which requires that all the singular values concentrate at 1. In fact, with the common practice of Gaussian weight initialization, isometry can never be achieved regardless of the choice of activation functions ([8]). To address such an issue, orthogonal weight initialization has been extensively studied in the past few years. For linear networks with arbitrary depth, orthogonality of each composing layers trivially leads to an isometry of the input-output Jacobian, and the benefit over Gaussian initialization in terms of training efficiency has been empirically observed ([12]) and theoretically justified ([5]).

Similar to weight initialization and regularization, next important component in isometric learning is non-linear activation. Now we will discuss a few related work with non-linear activation functions. In the past few years, many important variants of non-linear activation functions were developed that can obtain results closer to isometry. In ReLU, negative input values give rise to zero gradients and dead neurons that prevent effective training of the network. Motivating from the issue of dead neurons in ReLU, non-linear

activation functions like Leaky ReLU ([7]), Parametric ReLU ([4]), and Randomised ReLU ([13]) were designed to improve training of network. Motivated from the same idea Clevert et al. in 2016 ([2]) proposed Exponential Linear Unit (ELU). Another important motivation for ELU is that it pushes the mean of the activation closer to zero, therefore alleviates the issue with regular ReLU that the bias of the input signal is always shifted towards positivity. In isometric learning work by Qi et al. ([10]), SReLU was adopted which is highly related to ELU family and was first captured in the paper by Clevert et al ([2]). To bring our network close to an isometry SReLU should be preferred over ReLU, because SReLU lie between ReLU and identity, and hence can be considered as a trade-off between obtaining isometry and non-linearity.

2.2 In Context of U-NET

Since we have to create ISO-UNet, therefore it is important for us to understand UNet. U-net is an architecture used to perform semantic segmentation on images. A good thing about U-net is that it can perform semantic segmentation on a small training data set using data augmentation yielding good accuracy. It's a U-shaped architecture based on the idea of fully convolutional networks and comprises two paths, namely {i}contracting path (left) which uses convolution and pooling layers to reduce dimension and extract information from the input image, and {ii} expansive path (right) which uses convolution and transposed convolutions operation to use the extracted features and restore the image resolution as shown in figure 1 [11].

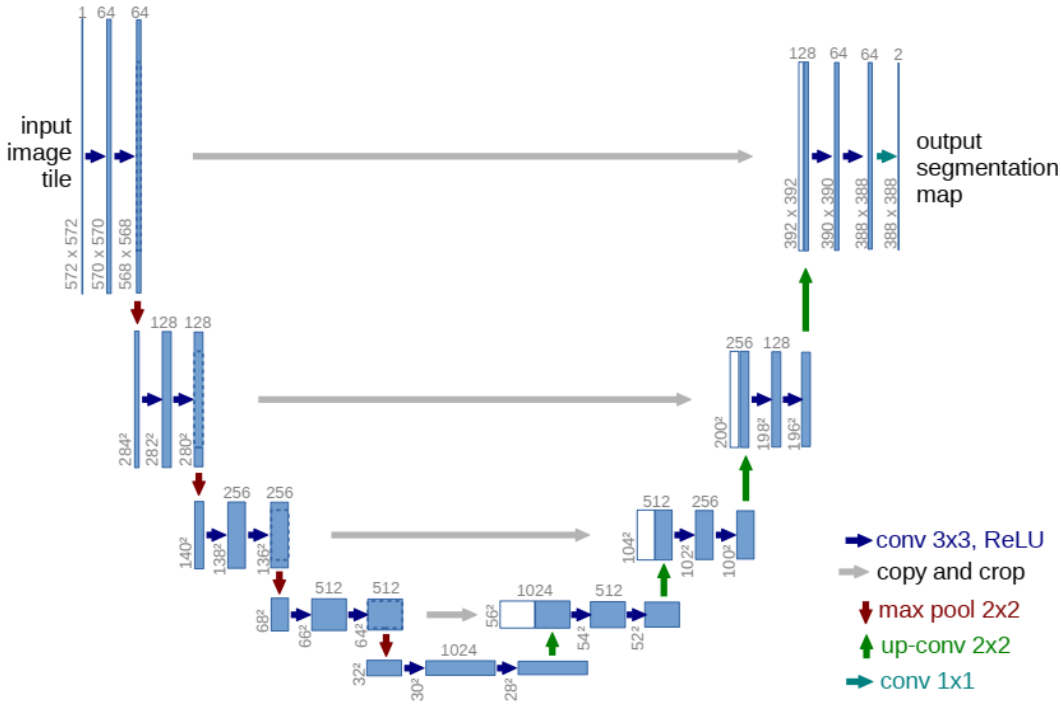

Figure 1: U-net architecture used for $572 \times 572 \times 1$ input image (adopted via [11]).

Let us now understand the working of U-net by taking example shown in fig 1. We will call two succes-

sive 3×3 convolution operations (padding = 0, stride = 1) followed by ReLU (represented by blue boxes) on an input as a convolution block, so in particular the U-net architecture comprises 9 convolution blocks. The convolution operation related to down sampling path have $64 * i$ number of kernels till $i = 5$ (where i is the number of convolution block), and in each upsampling convolution block, number of kernels reduce by half w.r.t. the output of previous convolution block. For the case of down sampling path the output of each convolution block passed through a 2×2 max pooling operation with stride 2 to get the contracted feature map, whose dimension (height$\times$width) is half compared to input of previous convolution block, and for the case of upsampling path the output of previous convolution block passes through a 2×2 transposed convolution operation to get an upsampled feature map, whose dimension (height$\times$width) is double the dimension compared to input of previous convolution block. The light grey arrows which runs from down-sampling path (left) to up-sampling path (right) corresponds to a skip-connection which is identical to identity operation and helps in extracting high resolution features from the corresponding down-sampling path feature map and is used as an input (concatenated with upsampled feature map) for the corresponding up-sampling layer. The skip connection operation here enhances the accuracy of the overall model by passing spatial information much deeper into the network. And finally, the prediction can be obtained by applying 1×1 convolution operation followed by sigmoid operation on the output of the last convolution block.

3 Methods and Approaches

This section is the heart of this project, here we will introduce the isometric principle and ISONet. Firstly we will introduce the concept of isometry in a vanilla ConvNet setup, and then move on to how can we enforce isometry in convolution kernels while training and initialization. Next we will see that how to design a non-linear activation function in isometric learning setup, and towards the end we will see that we can add skip connections or residual connections to ISONet to enhance its performance.

- **Isometric Learning:** Let $\mathcal{A} : \mathbb{R}^{C^{l-1}\times H\times W} \to \mathbb{R}^{C^{l}\times H\times W}$ denotes a convolution operation, $\phi(.)$ denotes a nonlinear activation function, and $x^0 : \mathbb{R}^{C^{0}\times H\times W}$ be the input signal. Then we will create a vanilla convolution network, which consists of only convolution operation and non-linear activation i.e.,

  $$x^l = \phi(y^l), \quad y^l = \mathcal{A}^l x^{l-1}, \quad l = 1, 2, ..., L$$

  If we assume that this ConvNet is constructed using squared loss (Loss $= \frac{1}{2}||z - x^L||_2^2$), then using backpropagation algorithm the differentiation of loss w.r.t. x^0 is given by:

  $$\frac{\partial Loss}{\partial x^0} = (\mathcal{A}^1)^*\mathcal{D}^1...(\mathcal{A}^L)^*\mathcal{D}^L(z - x^L)$$

  where $\mathcal{A}^* : \mathbb{R}^{C^{l}\times H\times W} \to \mathbb{R}^{C^{l-1}\times H\times W}$ is the adjoining operator of $\mathcal{A}^l$, and $\mathcal{D}^l : \mathbb{R}^{C^{l}\times H\times W} \to \mathbb{R}^{C^{l}\times H\times W}$ is the point-wise multiplication of $\phi'(y_{c,h,w})$ by l-th layer's $(c, h, w)^{th}$ input entry. Now we are ready to define the definition of isometry.
  Isometry: A map $\mathcal{A} : \mathbb{R}^C \to \mathbb{R}^M$ is called an isometry $\forall\{x, x'\} \subset \mathbb{R}^C$ if:

  $$< \mathcal{A}x, \mathcal{A}x' >=< x, x' >$$

  ISONet was designed to maintain isometry in the forward dynamic through operations $\mathcal{A}$ and ϕ, and in the backward direction through operations $\mathcal{A}^*$ and $\mathcal{D}$.
  .
  .

- **Isometry in Convolution:**
  Let

$$\mathbf{A} = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,C} \\ \alpha_{2,1} & \alpha_{2,2} & \cdots & \alpha_{2,C} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{M,1} & \alpha_{M,2} & \cdots & \alpha_{M,C} \end{pmatrix} \in \mathbb{R}^{M\times C\times k\times k} \tag{1}$$

  be a collection of convolution kernels, where each $\alpha_{m,c}$ is a kernel of the size $k \times k$ for $c \in \{1, ..., C\}$ and $m \in \{1, ..., M\}$

  Theorem 1: Given a convolutional kernel $\mathbf{A} \in \mathbb{R}^{M\times C\times k\times k}$ in eq (1), $\star$ a convolution operation, the operator $\mathcal{A}$ is an isometry if and only if:

$$\sum_{m=1}^{M} \alpha_{mc} \star \alpha_{mc'} = \begin{cases} \delta, & \text{if } c = c' \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

  and the operator $\mathcal{A}^*$is an isometry if and only if:

$$\sum_{c=1}^{C} \alpha_{mc} \star \alpha_{m'c} = \begin{cases} \delta, & \text{if } m = m' \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

  In above, δ is the Kronecker delta function defined on $Z \times Z$ that takes value 1 at coordinate (0, 0) and 0 otherwise.

  Delta kernel: We refer to a convolution kernel $\mathbf{A} \in \mathbb{R}^{M\times C\times k\times k}$ the (Kronecker) delta kernel, denoted as $\delta^{C\times M\times k\times k}$, if its entries indexed by (i, i, k_0, k_0) (assuming $k = 2k_0 + 1$) for all $i \in \{1, ..., min(C, M)\}$ are set to 1 and all other entries are set to zero. If $M \geq C$ (resp., $M \leq C$), then the operator $\mathcal{A}$ (resp., $\mathcal{A}^*$) associated with a delta kernel is an isometry. Moreover, if $M = C$ then a delta kernel is orthogonal.

  For practice, we may initialize all convolution kernels to be isometric by setting them to be the delta kernel, which we refer to as the Delta initialization.

  Now, coming to the part where we enforce kernels to orthogonality while training. For this, consider

$$Conv(\mathcal{A}, \mathcal{A}) = (\mathcal{A}(\alpha_{11}, ..., \alpha_{1,c}), ..., \mathcal{A}(\alpha_{M1}, ..., \alpha_{M,c}))$$

$$= \begin{pmatrix} \sum_{c=1}^{C} \alpha_{1c} \star \alpha_{1c} & \sum_{c=1}^{C} \alpha_{2c} \star \alpha_{1c} & \cdots & \sum_{c=1}^{C} \alpha_{Mc} \star \alpha_{1c} \\ \sum_{c=1}^{C} \alpha_{1c} \star \alpha_{2c} & \sum_{c=1}^{C} \alpha_{2c} \star \alpha_{2c} & \cdots & \sum_{c=1}^{C} \alpha_{Mc} \star \alpha_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{c=1}^{C} \alpha_{1c} \star \alpha_{Mc} & \sum_{c=1}^{C} \alpha_{2c} \star \alpha_{Mc} & \cdots & \sum_{c=1}^{C} \alpha_{Mc} \star \alpha_{Mc} \end{pmatrix} \tag{4}$$

  Now, comparing equation (4) with condition (3) we see that, $\mathcal{A}$ is isometric if and only if Conv($\mathcal{A}^T$, $\mathcal{A}^T$) = $\delta^{C\times C\times k\times k}$. Similarly, $\mathcal{A}^*$ is isometric if and only if Conv($\mathcal{A}$, $\mathcal{A}$) = $\delta^{M\times M\times k\times k}$. From these results we can deduce that isometry in convolutional kernels can be enforced by adding any one L($\mathcal{A}$) (if $C > M$) or $L(\mathcal{A}^T)$ (if $M > C$) to the objective function:

$$L(A) = \frac{\gamma}{2} \|Conv(\mathcal{A}, \mathcal{A}) - \delta^{M\times M\times k\times k}\|_F^2 \tag{5}$$

$$L(A^T) = \frac{\gamma}{2} \|Conv(\mathcal{A}^T, \mathcal{A}^T) - \delta^{M\times M\times k\times k}\|_F^2 \tag{6}$$

  where γ is the regularization coefficient.

- **Isometry in Nonlinear Activation:** ReLu is by far the most popular activation function to train deep neural networks. But since, the input signal using ReLu is normalized to zero mean and unit variance, isometry cannot be achieved using ReLU function.
  Trade-off between Isometry and Non-Linearity: It is difficult to achieve isometry and non-linearity together. By Mazur-Ulam theorem, any surjective isometry between two normed spaces over $\mathbb{R}$ is linear up to a translation. Therefore, isometry can never be achieved in the presence of nonlinearity . On the other hand, a purely isometric network without nonlinearity is not interesting either, since the entire network becomes a linear transformation which has very limited learning ability. The design of the nonlinear activation function requires striking a good trade-off between isometry and nonlinearity. For this, the authors advocated Shifted ReLU and tried to achieve a trade-off such that it should lie between isometry and non-linearity.
  SReLU is obtained by interpolating regular ReLU (for obtaining nonlinearity) and the identity (for obtaining isometry).Given a signal $y \in \mathcal{R}^{CHW}$ where C is the number of channels, SReLU is defined point-wise as:
  $$\phi_b(y) = max(y, b) \tag{7}$$
  where b is a parameter that is shared for each channel of y. If $b = 0$, SReLU becomes the regular ReLU. If $b \to -\inf$, it can be interpreted as a trade-off between nonlinearity and isometry. For all the experiments carried out, b is initialized to -1 and optimized in the training process.

- **Isometry in Residual Structure:** Here, the core idea is to introduce a skip connection to ISONet to improve its performance. We call the ISONet with residual connection as R-ISONet. The R-ISONet network learns a residual component that is added onto the path of signal propagation. If the residual component is small enough relative to the identity shortcut, such a network architecture is automatically near isometric i.e. skip connections help enforce isometry. Motivated by this, a scalar multiplier s called 'shared scale' is added at the end of each residual branch and initialize s to be zero (s is shared within each channel).

3.1 Work done before mid-term project review

Before mid-term review, we investigated the theoretical aspects of isometric learning involving isometry at initialization, isometry during training, isometry in non-linear activation functions and read up about the past research work done in this area and how isometry improves the network performance. However, none of the previous work had clearly demonstrated that the isometry property alone ensures surprisingly strong performance for deep networks.

We also carried out empirical investigation to verify the claims made in the paper. The details of the experiments are as follows:
Experiment 1:

- **Model**: ISONET
- **Dataset**: CIFAR-10 – 50000 train images, 10000 test images, No. of classes = 10
- **Data Transformations**: For training set- Random Horizontal Flip, Normalization, for validation set: Resize, Normalization
- **Loss criterion:** Cross Entropy
- **Loss Optimizer:** SGD
- **Weight Decay**- 1e-4, mom- 0.9

- **Learning rate** - Initial LR -ISONet - 0.02, Factor- 0.1, Epoch- 30,60,90
- **No. of epochs** – 100
- Regularization coefficient = γ

Results:

Comparison on training data		
Accuracy↓	For $\gamma = 0.0003$	For $\gamma = 0.0001$
Train Accuracy	92.23	95.20
Test Accuracy	87.28	90.32

Experiment 2:

- **Model**: R-ISONET
- **Dataset**: CIFAR-10 – 50000 train images, 10000 test images, No. of classes = 10
- **Data Transformations**: For training set- Random Horizontal Flip, Normalization, for validation set: Resize, Normalization
- **Loss criterion:** Cross Entropy
- **Loss Optimizer:** SGD
- **Weight Decay**- 1e-4, momentum- 0.9
- **Learning rate** - Initial LR -ISONet - 0.02, Factor- 0.1, Epoch- 30,60,90
- Regularization coefficient = γ
- **No. of epochs** – 100

Results:

Comparison on training data		
Accuracy↓	For $\gamma = 0.0003$	For $\gamma = 0.0001$
Train Accuracy	93.9	97.308
Test Accuracy	89.64	93.86

Experiment 3:

- **Model**: ISONET
- **Dataset**: CIFAR-100 – 50000 train images, 10000 test images, No. of classes = 100
- **Data Transformations**: For training set- Random Horizontal Flip, Normalization, for validation set: Resize, Normalization
- **Loss criterion:** Cross Entropy
- **Loss Optimizer:** SGD
- **Weight Decay**- 1e-4, momentum- 0.9

- **Learning rate** - Initial LR -ISONet - 0.02, Factor- 0.1, Epoch- 30,60,90

- **No. of epochs** – 100

Results: ISONet

Comparison on training data		
Accuracy↓	$\gamma = 0.0003$	$\gamma = 0.0001$
Train Accuracy	72.9	76.108
Test Accuracy	62.17	69.186

Results: R-ISONet

Comparison on training data		
Accuracy↓	$\gamma = 0.0003$	$\gamma = 0.0001$
Train Accuracy	78.90	85.438
Test Accuracy	69.25	77.10

3.2 Work done after mid-term project review

Major comments that we received on our mid-sem project presentation includes replication of object detection task using COCO data set, and construction of ISO-UNet for semantic segmentation task. Experiments for object detection using COCO dataset was suggested, to verify the superior results claim made in the paper, for which no official code was provided.

For object detection experiment, we constructed a Faster-RCNN framework with R-ISONet as the backbone network. The classifier head of the R-ISONet was removed to extract feature maps of size 16X16X2048. These feature maps are then passed to the Region proposal network which generates region of interests proposals using anchors. These Region of Interest (RoIs) are pooled with the shared feature maps from the backbone network and are input to the classifier, which gives bounding box predictions, class labels and probabilities as the final output.

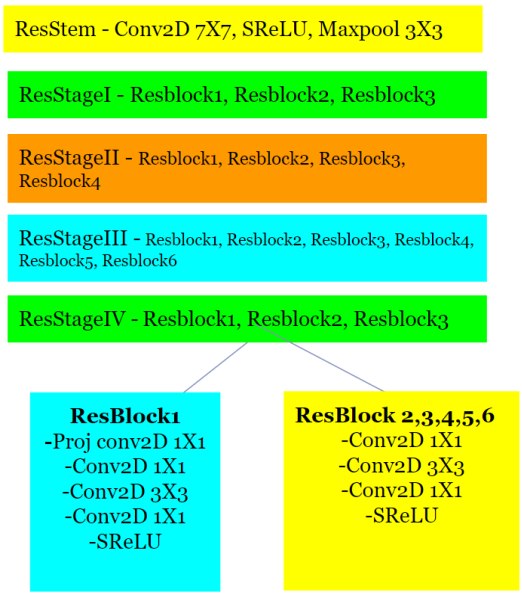

Figure 2: 50-layer R-ISONet Architecture

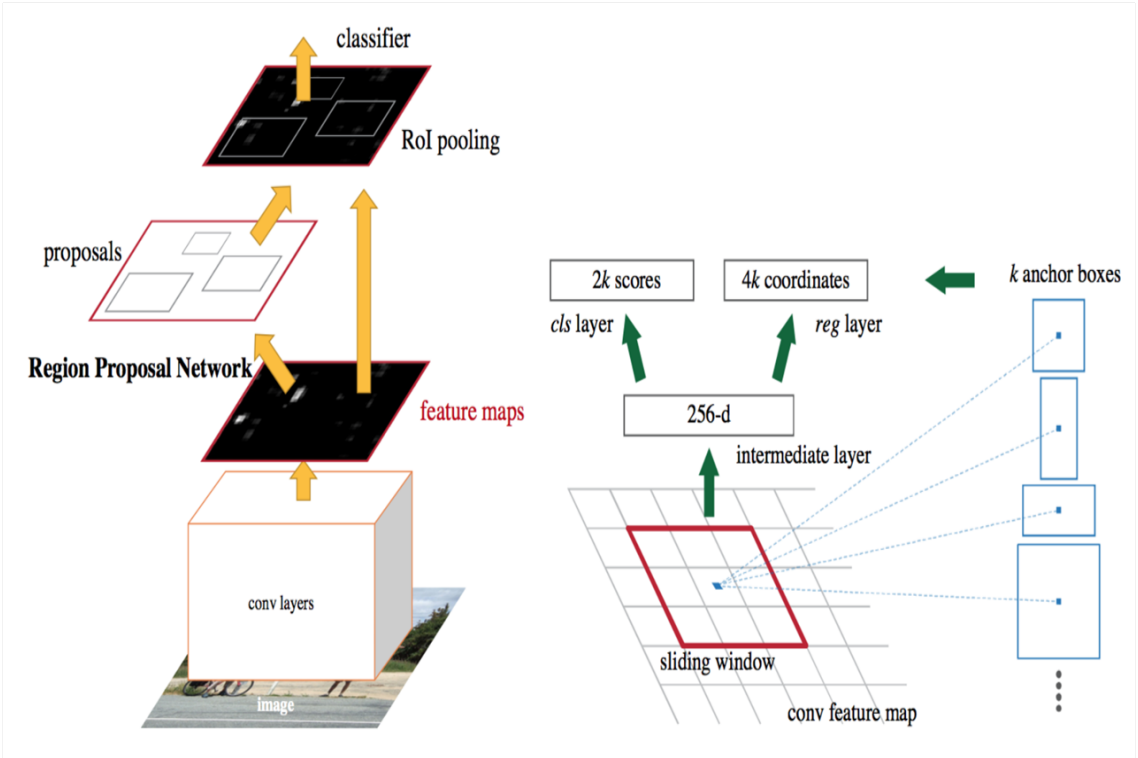

Figure 3: Faster-RCNN Architecture

Now, we will discuss the construction of ISO-UNet for image segmentation task. We have already explained UNet in section 2 and isometric learning in section 3. We combined these two things to construct ISO-UNet. The convolution block for ISO-UNet consists of two successive 3×3 convolution operations (padding $= 1$, stride $= 1$) followed by $SReLU$ non linear activation. The output of this convolution block was passed to a Maxpool layer with stride 2, and its output acts as input to the next convolution block. We have 5 such convolution blocks which made our encoder. The output of encoder was then fed to the deconvolution block. The deconvolution block consists of a transposed convolution operation (padding $=$ 2, stride $= 2$) followed by the convolution block. NOTE: The output of transposed convolution operation in deconvolution block have to first concatenate with the output of $(5 - i)^{th}$ convolution block in encoder (where i denotes the deconvolution block number staring from 1 which is next to encoder), and this concatenated tensor then passes through the convolution block of deconvolution. We have 4 such deconvolution blocks which made our decoder. The output of decoder then passes through a 1×1 convolution operation to give final output. The kernels were initialized to be same as Kronecker delta kernels mentioned in [10] and were enforced to near orthogonal during training. This concludes the construction of our ISO-UNet. We performed experiments on ISO-UNet using brain MRI data set and found that it performs on par with UNet [10], more details of this can be found in section 5 and 6.

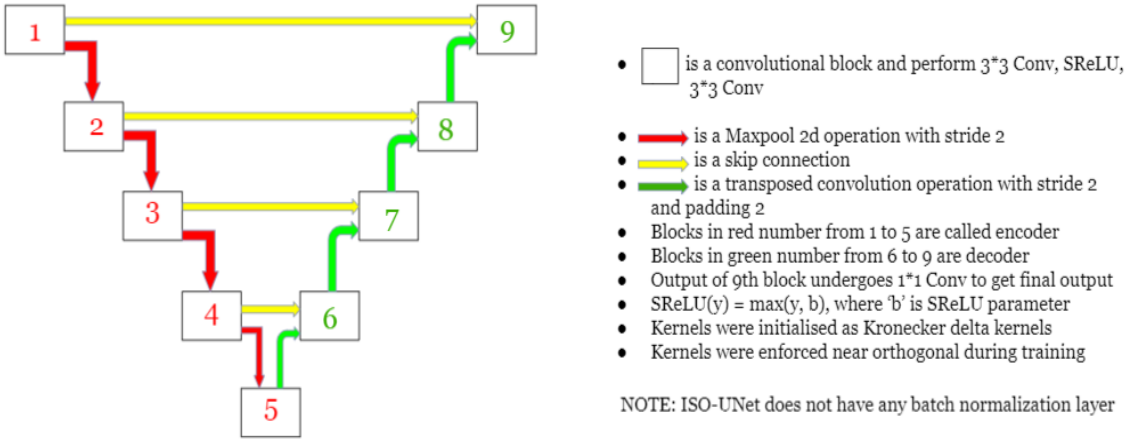

Figure 4: Summary of ISO-UNet architecture in a picture.

4 Data set Details

4.1 For ISO-UNet

Data set : The data used in this project was downloaded from `https://www.kaggle.com/mateuszbuda/lgg-mri-segmentation`, and adopted from [1] which was obtained from The Cancer Genome Atlas (TCGA) and The Cancer Imaging Archive (TCIA). The data consist of 3929 brain MRI images along with their tumor mask from 110 patients. The task is to detect cancer tumor in the image, the mask is a grayscale image where black represents the background and white part(s) represent the detected tumor (see fig 5). Out of 3929 images 1373 images are those whose mask is complete black, which means there is no tumor in the brain.

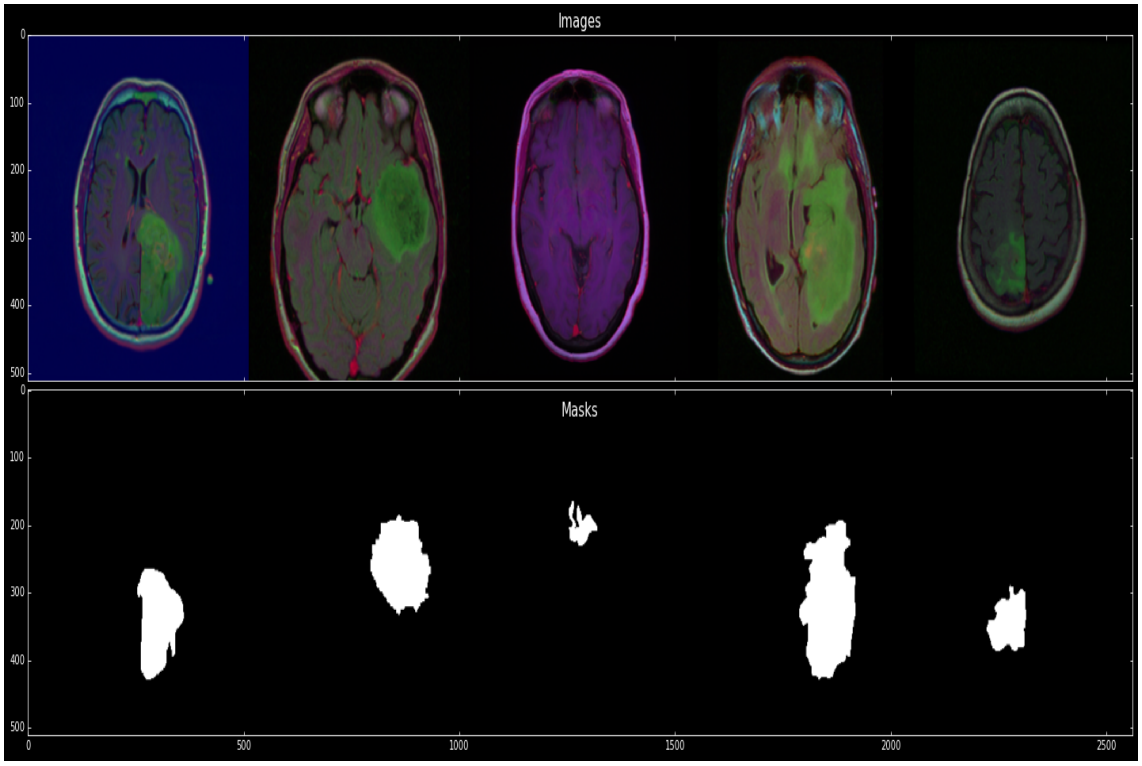

Figure 5: Brain MRI images and their corresponding masks.

Data Preprocessing : Since the images are all different for different patient, their size also varies. So the first step in preprocessing these images and their masks was to crop them to the smallest enclosing of skull image. After cropping, padding (with zero) was done if the image was not square; otherwise no padding was done. The next step was resizing these images along with their masks to $224 \times 224 \times \#channels$ (in case of input, #channels = 3; and for mask, #channels = 1), since after padding there is a good chance that these images may vary in size. And the final step was channel-wise standard normalisation of these images.

4.2 For object detection using R-ISONet

Data set : The annotations file used in this project was downloaded from `https://cocodataset.org/`, and the images were downloaded to using 'coco-url' provided for each image ID in the annotation file . The entire dataset consists of 118K real-world images in the train set and 5K test images. Given the huge dataset size and the associated memory limitations in Colab, we were unable to train the network on the entire dataset. We have constructed a subset of the COCO dataset with just 3 classes- person, car, bicycle. The subset consists of 1K train images and 55 test images. To access the data, pycocotools API is used

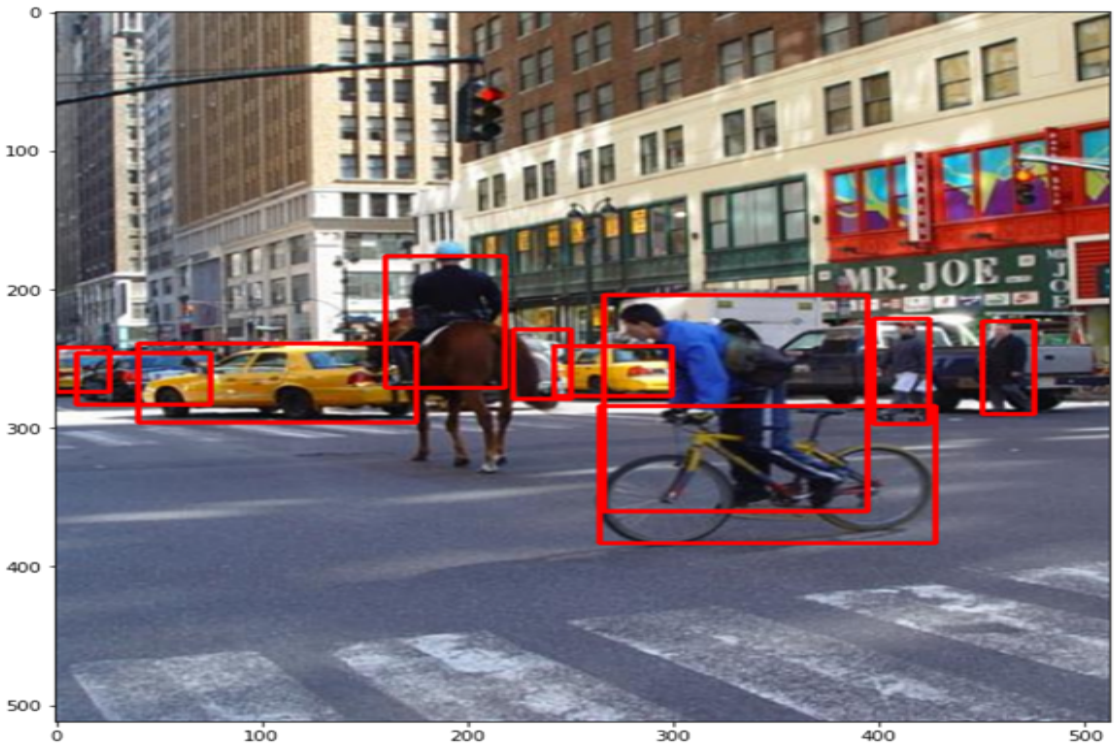

Figure 6: A picture from COCO dataset with ground truth bounding boxes

Data Preprocessing : Again, since the images in the dataset are all of different sizes, the images are resized to (512X512X3). Annotations were converted from (xywh) format to (xyxy) format. The bounding box coordinates were adjusted too, according to the resized images. For this Abulmenations API was used. Other pre-processing involved normalization and tensorization of the data.

5 Experiments

In this section we will discuss details related to our experiments on ISO-UNet and R-ISONet for object detection. Kindly note that these experiments were conducted on Google Colab environment which provides a single 12GB NVIDIA Tesla K80 GPU.

5.1 For ISO-UNet

The constructed ISO-UNet (see section 3 or details) was trained on a brain MRI data set. The SReLU parameter b was fixed to -1 similar to [10]. We divided the data into two parts i.e. 85% into training and 15% into validation, and a batch size of 20 for both training and validation data was used. We trained our model using a SGD optimizer with learning rate = 0.025, momentum = 0.9 and weight decay of $1e-4$. The regularization coefficient (γ) we used was $1e-4$. We trained the model for 50 epochs, and the learning rate was decreased by half every $5i$ epoch (for $i = 1, 3, 5, 7, 9$).

To measure performance of our ISO-UNet, we used two metrics i.e., mean Dice loss to calculate loss and

mean intersection over union (mIoU) to measure accuracy. The Dice loss is simply $1-$**Dice coefficient**, where Dice coefficient measures the overlap between two samples, its values lies between $[0, 1]$ where 1 implies a perfect overlap. The dice coefficient is given by :

$$Dice = \{2 * \sum_{i=1}^{N}(P_i * Q_i)\} / \sum_{i=1}^{N}(P_i^2 + Q_i^2)$$

where P represents the prediction matrix, Q represents the true label matrix and N denotes total number of pixels. The overall loss is given by taking mean of Dice loss obtained for each training instance. We measure the accuracy by calculating mIoU, it is a standard performance measure for image segmentation tasks. For a given set of images, IoU calculates the similarity between the ground truth mask and predicted mask for the object present in the image. The IoU can be calculated as :

$$IoU = \{\sum_{i=1}^{N} P_i * Q_i\} / \sum_{i=1}^{N}\{P_i + Q_i - P_i * Q_i\}$$

where P represents the prediction matrix, Q represents the true label matrix and N denotes total number of pixels. The overall accuracy is given by taking mean of IoU obtained for each training instance.

5.2 For object detection using R-ISONet

Since official code for these experiments was not made available by the authors, it was not clear if they used pre-trained weights or not. We constructed four frameworks with varying backbones to Faster-RCNN classifier to understand the transfer capabilities of the R-ISONet, as claimed in the paper. First, we constructed a framework with R-ISONet + Faster RCNN and trained it from scratch. The results were compared with another framework ResNet50 + FPN + Faster-RCNN without any pre-trained weights, which was also trained from scratch. Next, we introduced the ImageNet pre-trained weights to the backbone of both the networks and compared the results.

6 Results

6.1 For ISO-UNet

Following the details mentioned in section 5.1, the mean dice loss on training data we got on training data is 0.083 (approx) and the mean IoU (accuracy) % we got is $\sim$ 84.76%. On the validation data we got around 0.087 mean dice loss and 83.99 mean IoU (accuracy). We have compared our ISO-UNet with UNet on the same data set and the results we got are mentioned in table below, which clearly shows that our ISO-UNet is performing on par with UNet. Kindly note that we have compared our result with the UNet from here: `https://www.kaggle.com/monkira/brain-mri-segmentation-using-unet-keras`, also the number of epochs used in UNet are 150 while in ISO-UNet are 50 only.

Comparison on training data		
Network↓	Mean dice loss	Mean IoU(%)
ISO-UNet	0.083	84.76
UNet	**0.08**	**85.2**

Comparison on validation data		
Network↓	Mean dice loss	Mean IoU(%)
ISO-UNet	**0.087**	**83.99**
UNet	0.097	83.14

Some predictions on the images from validation data are:

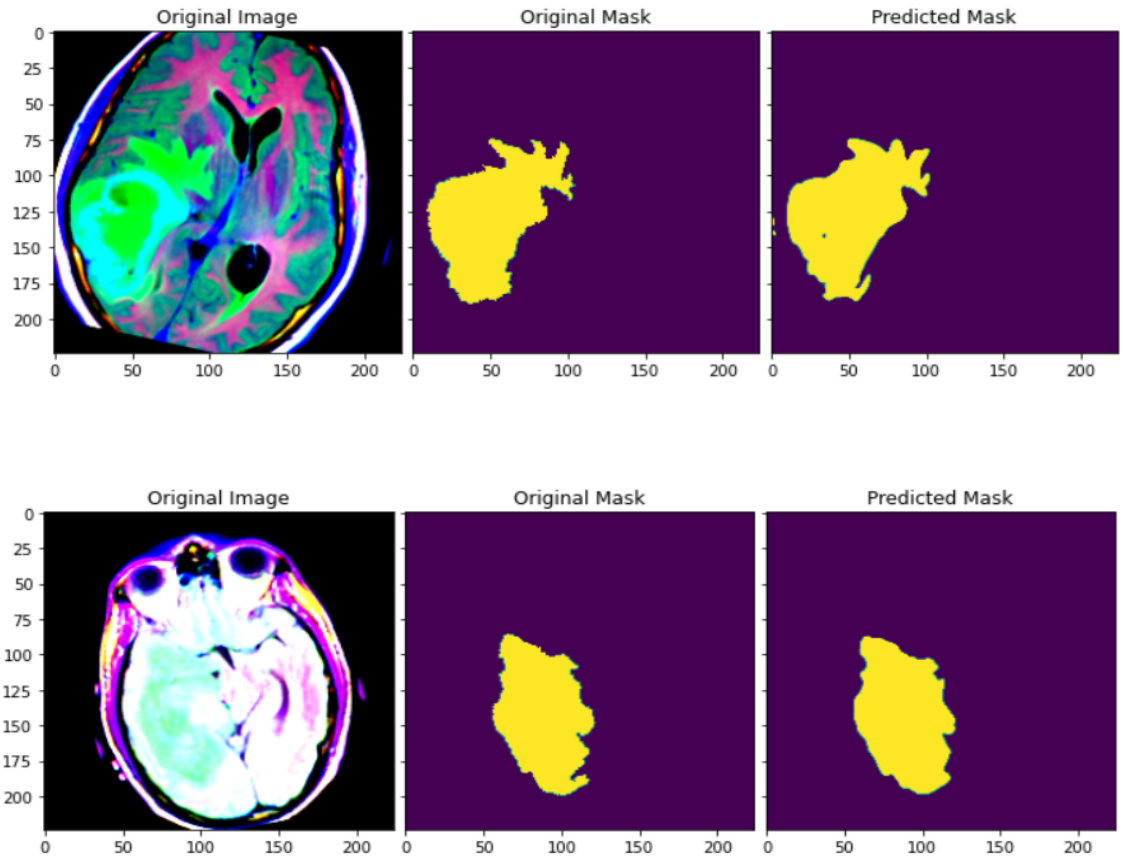

Figure 7: Brain MRI samples along with their corresponding and predicted masks.

6.2 For R-ISONet + Faster-RCNN

The results obtained using all 4 frameworks are as follows:

Comparison between R-ISONet and ResNet		
	R-ISONet 50 + Faster RCNN	ResNet 50 (with FPN) + Faster RCNN
Without pretraining	mAP @ IoU 0.50:0.95: **27.9**, mAP @ IoU 0.75: **34.8**	mAP @ IoU 0.5:**26.5**, mAP @ IoU 0.75:**34.5**
With pretrained weights	mAP @ IoU 0.50:0.95: **33.2**, mAP @ IoU 0.75: **35.6**	mAP @ IoU 0.50:0.95: **43.5**, mAP @ IoU 0.75: **46.3**

It can be clearly seen, during training from scratch, that the R-ISONet results are better than Resnet + FPN backbone, thus corroborating the results in the paper.

Note: The mAP results are not replication of the ones mentioned in the paper. This is because of training on a small dataset due to Colab memory limitations on training huge datasets. The presented results, however depict the key result- superiority and ease of training of isometric networks and at par performance with complicated networks.

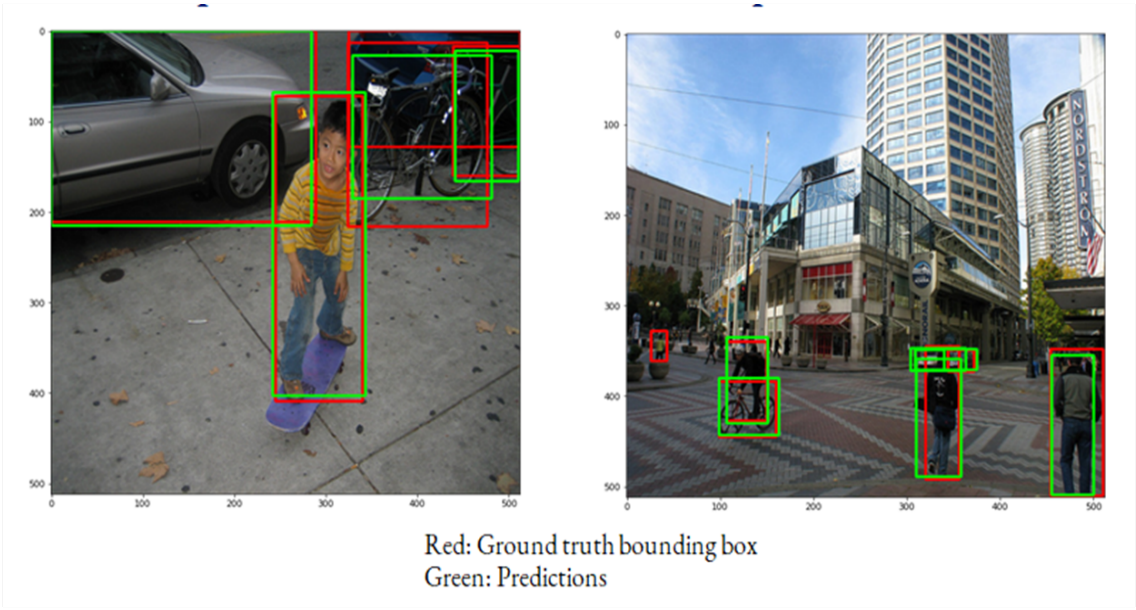

Figure 8: Predictions with R-ISONet vs ResNet

7 Future Work

- We have seen that isometric learning is giving excellent performance on task related to image classification, segmentation and object detection using convolutional neural networks. Therefore we are suggesting to use isometric learning in recurrent neural networks for natural language processing task as well, because batch normalization and initialization is also a common practice in RNN (and its variants). We can also use isometric learning in generative networks and autoencoders. In simple words, with isometric principle people can discover or design new simple architectures with improved performance in future.

- An interesting theoretical work in future can be to investigate the following question:
  Instead of enforcing isometry i.e. $< Ax, Ax' >=< x, x' >$, what will happen if we enforce $< Ax, Ax' >= c_l < x, x' >$ or $< Ax, Ax' >=< x, x' >_{M_l}$ (the inner product with respect to some matrix M_l denoted by $x^T.M_l.x'$)? Here the terms c_l and M_l are layer dependent parameters which can be tweaked according to the characteristics of the inputs to each layer. This future work was provided with reference to the comment we received in mid-sem review by instructors.

8 Conclusion

The world is changing at a faster rate than ever, and every now and then a new component for neural network architecture is being designed. In this project work we have seen that when deep ConvNets faces architecture design challenge (like initialization, normalization , etc.), then isometric learning becomes useful by acting as a central guiding principle for training very deep ConvNets. We have seen through experiments that ISONet, R-ISONet and ISO-UNet delivers performance that is on par with standard networks like ResNet and UNet. If we consider ISONet as a baseline model then a lot of other type of ISONets can be constructed for different tasks (just like we created ISO-UNet for semantic segmentation task). We can try to use isometric learning in natural language processing and other areas as well where normalisation and initialization were considered as time and memory consumption task.

References

[1] Mateusz Buda, Ashirbani Saha, and Maciej A Mazurowski. Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. *Computers in biology and medicine*, 109:218–225, 2019.

[2] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[5] Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. *arXiv preprint arXiv:2001.05992*, 2020.

[6] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[7] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer, 2013.

[8] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932. PMLR, 2018.

[9] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. *arXiv preprint arXiv:1606.05340*, 2016.

[10] Haozhi Qi, Chong You, Xiaolong Wang, Yi Ma, and Jitendra Malik. Deep isometric learning for visual recognition. In *International Conference on Machine Learning*, pages 7824–7835. PMLR, 2020.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[12] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[13] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.