UNIVERSITY OF TARTU
Institute of Computer Science
Data Science Curriculum

Joonas Ariva

Fast Fourier Convolutions in Self-Supervised Neural Networks for Image Denoising

Master's Thesis (15 ECTS)

Supervisor: Mikhail Papkov, MSc

Tartu 2022

Fast Fourier Convolutions in Self-Supervised Neural Networks for Image Denoising

Abstract: Quality of digital images depends on a multitude of environmental and equipment factors. In many cases our options for optimizing imaging conditions are limited, and the acquired images turn out to be corrupted with noise. Recently, denoising convolutional neural networks (CNN) have started to outperform classical denoising algorithms. If approached naively, these networks require a lot of pairs of noisy and clean images from the particular domain. In some fields (*e.g.* in biomedical imaging) it is hard to collect such data in abundance. This limitation has accelerated a research for self-supervised networks what can learn denoising just from noisy images alone. However, such networks' performance could be constrained by the the limited receptive field of regular convolution.

To mitigate this problem, a new modification for CNNs was proposed: *Fast Fourier Convolution* (FFC). Here, a global receptive field is achieved by using Fourier Transform and convolving spectral representation. Global perception field can help CNNs to better capture dependencies in image regions which are far apart. Given the ability of FFC to enhance multiple state-of-the-art classification neural networks, we hypothesize that denoising neural networks could also gain from its use.

In this work, we design multiple approaches for incorporating FFC into self-supervised neural networks for image denoising. We evaluate these approaches on three diverse benchmark datasets and compare them with both supervised and self-supervised methods. We empirically show that FFC-enhanced denoising network achieves the state-of-the-art results on character dataset and shows comparable level of performance for both grayscale and color natural images.

Keywords:
deep learning, neural networks

CERCS: T111 - Imaging, image processing; P176 - Artificial intelligence

Fourier' sidumid juhendamata mürapuhastuse tehisnärvivõrkudes

Lühikokkuvõte: Keskkond ja kasutatav tehnika mõjutavad kujutamisel loodud piltide kvaliteeti. Mitmetel juhtudel ei ole aga võimalik neid tegureid optimeerida ja saadud pildid on mürased. Hetkel põhinevad parimad mürapuhastusmeetodid sidumnärvivõrkudel (ingl. k. *Convolutional neural network* ehk CNN). Üldiselt vajavad sellised tehisnärvivõrgud hästi töötamiseks palju müravabu pilte valdkonnast. Osades valdkondades (näiteks biomeditsiiniline mikroskoopia) on selliseid pilte keeruline koguda suurtes kogustes. Seetõttu on muutunud oluliseks juhendamata õppe närvivõrgud, mis suudavad õppida mürapuhastust vaid müraste piltide pealt. Väikese nägemisväljaga sidumid võivad aga selliste närvivõrkude täpsust piirata.

Selle probleemi üks võimalik lahendus võib olla hiljuti välja pakutud Fourier sidum (ingl. k. *Fast Fourier Convolution* ehk FFC). Sellel sidumil on piiramatu nägemisväli, kuna enne sidumioperatsiooni viiakse läbi Fourier' pööre. Piiramatu nägemisväli võib aidata CNN-idel paremini seostada omavahel pildiosi, mis asuvad üksteisest kaugemal. FFC-ga on täiustatud mitmeid kaasaegseid klassifitseerimise närvivõrke. Magistritöö hüpotees on, et ka mürapuhastuse närvivõrgud võiksid FFC-st kasu saada.

Magistritöös disainitakse mitu erinevat meetodit, kuidas FFC-d juhendamata mürapuhastuse närvivõrkudes kasutada. Meetodeid testitakse kolmel eri võrdlusandmestikul ja tulemusi võrreldakse teiste mürapuhastusmeetoditega. Hiina kirja andmestikul saavutatakse FFC-ga täiustatud närvivõrkudega tipptasemel tulemused ning loomulike värviliste ja halltoonides piltide andmestikel saavutatakse olemasolevate meetoditega võrreldavad tulemused.

Märksõnad:
sügav õppimine, tehisnärvivõrgud

CERCS: T111 - Pilditehnika; P176 - Tehisintellekt

Contents

1 Introduction

Imaging is an important concept to better understand the world around us. However, the acquired images are rarely perfect copies of the imaged scene. Imperfections in the imaging system, non-ideal conditions, and environmental perturbations create unwanted modifications (noise) to the images, lowering their quality.

There are multiple ways to combat noise in imaging. For example, we can increase the image acquisition time, improve lighting conditions or apply post-processing algorithms. Still, some of these methods might not be viable options when cutting-edge science is concerned.

For instance, in microscopy imaging in biology, there are situations where decreasing the light exposure could be favorable. There can be many reasons for that: speeding up the measurements, avoiding damaging fragile samples with excessive laser light, or spending the photon budget for better spatial resolution instead. In such cases, we can only rely on post-processing methods for image denoising.

Over the years, various analytical algorithms have been proposed for image denoising [1][2]. However, thanks to fast advancements in machine learning, convolutional neural networks (CNN) have started to outperform classical algorithms. One limiting factor of supervised CNNs is that they require lots of training data to work well. For denoising the training data comprises of pairs of noisy images and their respective noise-free analogues. In many fields collecting large amounts of clean samples is not viable and so there has been a push for self-supervised networks that can learn denoising from noisy images alone.

In 2020 a new modification for CNNs was proposed: *Fast Fourier Convolution* (FFC) [3]. The motivation for this proposal stems from the fact that regular convolutions have a limited visual field of perception and might fail to capture essential details of the image that are far apart. This problem is solved using FFC, as it operates in the spectral domain and "sees" the whole image all at once. These FFC blocks can be imputed in place of vanilla convolutions. The authors of FFC demonstrate that multiple state-of-the-art neural networks could benefit from the use of FFC. We hypothesize that denoising neural networks could also gain from the use of FFC.

The goal of the master thesis is to investigate the effect of FFC on self-supervised denoising neural networks. We combine FFC with state-of-the-art self-supervised denoising method *Noise2Same* [4]. We test different strategies to integrate FFC blocks into encoder-decoder CNNs on different datasets (BSD68, Hànzì, ImageNet). Furthermore, we investigate how FFC hyperparameters affect denoising performance. To our best knowledge, this is the first attempt to utilize FFC in denoising neural networks.

In the first chapter, we give an overview of image denoising, relevant neural networks, and FFC. In the second chapter, we describe network designs and experimental setup. In the third chapter, we discuss the experimental results. Finally, in the last chapter, we summarize the thesis content.

2 Background

In this chapter we give an overview of digital image denoising, artificial neural networks and *Fast Fourier Convolution*.

2.1 Denoising

Denoising is the process of recovering a clean signal from its noisy acquisition. In digital imaging, almost always, some sort of noise is added to the "true" image, degrading its quality. This means that the pixel values of the picture might not represent the imaged scene truthfully. Noise can be caused by multiple reasons: insufficient lighting, unwanted background lightning, aberrations from an optical system, or limitations from the electronic components.

2.1.1 Noise types in digital imaging

Here we give a short overview of the main types of noise one can encounter in digital imaging. These include Gaussian noise, salt-and-pepper noise (both Figure 1) and shot noise (Figure 2).

Gaussian noise (Figure 1b) can manifest from many different physical processes, which have random elements. The root cause comes from the central limit theorem: when independent random variables are summed up, their sum tends toward normal distribution. Thus the added noise values for each pixel are drawn from the normal distribution (1), where μ is the mean value of the noise and σ^2 is the variance.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{1}$$

Salt-and-pepper noise (also called impulse noise; Figure 1c) comes from errors in data transmission and failures of electronic components. This noise takes the form of white and black pixels in the image, which looks like somebody has shaken salt and pepper to the image. Salt-and-pepper noise can be modeled with a Bernoulli distribution.

Shot noise (also called Poisson noise) is a type of noise, which can be modelled by a Poisson process. In digital imaging, shot noise occurs during photon counting in an optical sensor. The number of photons detected by the sensor at any given time fluctuates creating noise in the image. This happens due to photons' independence to each other, meaning that they arrive at random times. Arrival of photons over a time interval t can be described with a discrete Poisson distribution:

$$P(N = k) = \frac{(\lambda t)^k e^{-\lambda t}}{k!}, \tag{2}$$

(a) Original image (b) Gaussian noise (c) Salt and Pepper noise

Figure 1. Example of Gaussian and Salt and Pepper noise.

Figure 2. Shot noise simulation. Light exposure is increased from left to right.

where N is the amount of photons measured and λ is the expected number of photons per time unit t (proportional to the illuminance of the scene). Photon arrival time fluctuations are evened out if the exposure to the light is high or the measurement time is long. To the contrary, shot noise becomes very apparent when imaging in low light conditions.

2.1.2 Classical denoising algorithms

Usually, denoising is an ill-posed problem, meaning that there is not enough information in the noisy picture for perfect reconstruction. However, we can attempt to restore the original image if we make assumptions about the type of noise in the picture. Useful information about the noise includes its nature and its power spectral density. We can also make assumptions about image structure and use its self-similarity for restoration purposes.

Different statistical methods have been developed on these sets of assumptions to denoise digital images. Simpler methods include median and average filters, which

assign each pixel average or median value of its neighbours, respectively. Averaging filters are good if the noise values are independent of each other and have a zero mean (noise can be averaged out with multiple acquisitions or longer exposure time). Median filters help to combat extreme pixel values like the ones that salt-and-pepper noise adds. However, simply aggregating over neighboring pixel values harms the sharpness of the image. More sophisticated denoising algorithms include Block Matching and 3D filtering [2] and Non-local means [1]. These methods are commonly used as benchmarks when evaluating new denoising algorithms.

Non-local means (NLM) uses averaging of pixel values, but in a more complex way than an average filter. Pixels are weighted over their similarity, and averaging happens over the most similar pixels, which might not be located in the target pixel's immediate neighbourhood. Pixel-wise implementation of non-local means for denoising a color image $u = (u_1, u_2, u_3)$ at pixel p follows as:

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q) w(p,q), \tag{3}$$

where $i \in \{1, 2, 3\}$ and $B(p, r)$ marks a neighborhood centered around pixel p with a size of $(2r + 1) \times (2r + 1)$ pixels. The choice of patch radius r depends on the standard deviation of noise: for higher values of σ, larger patches are needed for denoising. The weighting function $w(p, q)$ calculates the similarity between a patch $B(p, r)$ and $B(q, r)$. Squared Euclidean distance is used for the similarity metric. $C(p)$ is a normalization term: $C(p) = \sum_{q \in B(p,r)} w(p, q)$. We can see that only pixels in the range of $B(p, r)$ are considered for the algorithm, and hence the name "non-local means" is a bit misleading: it is still a semi-local algorithm (in theory, the range can be set to cover the whole image, but in practice, it would be too slow).

Block matching and 3D filtering (BM3D) also focuses on pixel neighbourhood similarity. The algorithm has three steps: block matching, collaborative filtering, and aggregation. At first, similar patches of an image are grouped together to form 3D arrays (Figure 3). Then, collaborative filtering happens: each stack is linearly transformed to another basis, Wiener filtering or a thresholding strategy is applied to reduce the noise, and finally, the stack is converted back to the original domain with an inverted transformation. Finally, the patches are moved back to their original position on an image. Some patches can overlap, so the final picture is obtained by aggregating all of the local (pixel) estimates using a weighted average.

Recently denoising neural networks have come into spotlight as they start to outperform classical denoising algorithms [5, 4, 6]. Hence, this thesis focuses on neural networks for denoising. Next, we will go over the theory on artificial neural networks and then discuss denoising with neural networks.

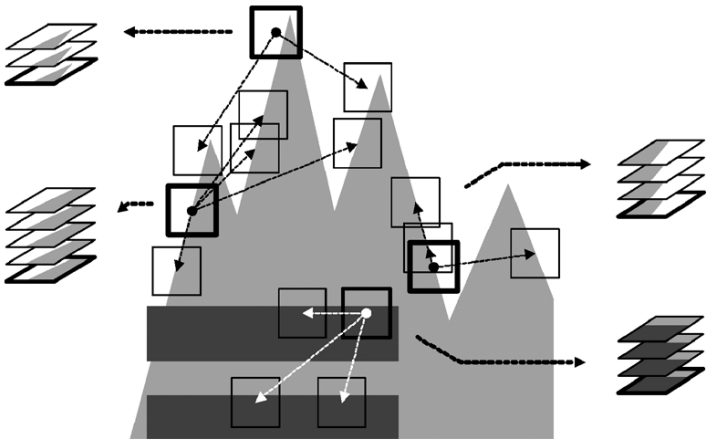

Figure 3. Example of block matching in BM3D: similar patches are stacked together [2].

2.2 Artificial neural networks

Artificial neural networks are a subclass of machine learning models inspired by biological neural networks in the brain. Neuron, the smallest information processing unit in the brain, "listens" to electrochemical signals from connected neurons. If the total input signal to the neuron reaches a certain threshold it will generate its own electrical signal. While the underlying biological process is very complex, this idea of a neuron as a computational unit is powerful and can be generalized for the purposes of computer science.

2.2.1 Perceptron

Single computational neuron is called perceptron and can be described with the function

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $\mathbf{w} \cdot \mathbf{x}$ is an inner product between a vector of inputs and vector of weights and b is a bias term. Essentially, if the sum of weighted inputs is larger than a threshold, perceptron outputs a 1 and otherwise a 0. This non-linearity of the perceptron is integral for neural networks to learn complex patterns from the data.

2.2.2 Fully Connected network

Multiple layers of perceptrons can be formed into a Fully Connected (FC) network. Each layer of perceptrons gets its inputs from all of the perceptrons in the previous layer and outputs to the perceptrons in the next layer (Figure 4). The first layer uses prediction features as an input, and the last layer of the network outputs the prediction(s). The threshold activation function of a perceptron is usually replaced by a more complex function such as Rectified Linear Unit (output scales linearly with the input if the threshold is reached) or sigmoid function.

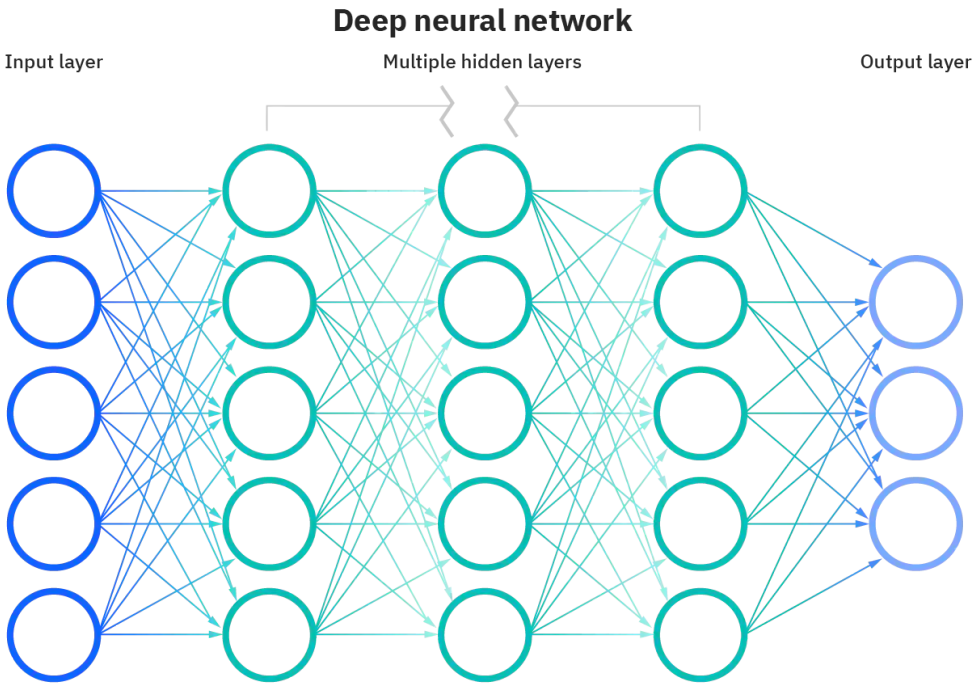

Figure 4. Example of fully connected neural network with multiple hidden layers [7].

Usually, the weights of the network are randomly initialized from a zero-mean Gaussian distribution or uniform distribution [8, 9]. And so, at first, the model outputs completely random predictions. Appropriate values for the weights need to be learned during the training. Network training needs a learning rule - a loss function - and an optimizer to act on the learning rule. After a batch of training data is passed through the network, the loss function measures the goodness of the predictions compared to the ground truth. Then partial derivatives of the loss function are calculated w.r.t each weight of the network using backpropagation algorithm [10]. Based on the derivatives, the optimizer changes the network weights to minimize the loss function (*e.g.*, mean squared

error for a regression task or cross-entropy for a classification task). After updating the weights, a new batch of data will be passed through the network, and the training process will repeat.

Because the distribution of training data might change from batch to batch a good practice is to add batch normalization layers between FC layers. Standardizing and affine rescaling of the inputs before each layer has been shown to accelerate the training and make the network more robust to weight initialization strategies [11].

Training data is iterated multiple times through the network. The goal of the training is to find such parameters (weights) for a model so that it could best fit the data distribution presented to it by minimizing the loss function.

2.2.3 Convolutional neural network

FC network is not the best solution for tasks related to visual data. Images have high dimensionality and therefore it would require huge FC networks to meaningfully learn from image data. For example, consider a relatively small color image of size 256×256 pixels. It would have a dimensionality of 3×256×256 = 196608. Besides the high dimensionality, we also have to acknowledge the importance of context. To understand what a pixel in an image is representing we cannot just look at the individual pixel — it is vital to consider the neighborhood of the pixel. Unfortunately, FC networks don't have any inherent local context sensitivity built into them. Convolutional Neural Networks (CNNs), inspired by human visual perception, attempt to solve both problems by using moving filters.

CNNs use layers of moving neurons (also called filters or kernels) that slide over the image and in each position use all of the underlying pixel values to output a single value to the next layer. Usually, small kernels (*e.g.*, 3×3 or 5×5) are used consecutively. Therefore the information about a pixel neighborhood is preserved through layers. It is also common to use 1×1 kernels to introduce channel interaction. The number of parameters in CNN is fixed with the choice of filters and is not tied to the resolution of the images. Essentially, these filters transform the image the same way as convolutions in signal processing. Usually, multiple convolutional layers are stacked together. The first layers extract low-level features from the image such as edges and corners. Deeper layers can then build upon it and extract much more complex features relevant to the task at hand. Each moving filter generates its own feature map (also called a channel) and one layer can have multiple filters (Figure 5). Dimensions of the feature map depend on the size of the input, size of the filter and the stride of the filter. Usually the number of feature maps increases with the depth of the network but the size of these maps decreases due to downsampling layers. For the classification task, the network converges to the feature representation and finishes with a fully connected layer, which predicts the label of the image. For the tasks, when the output of the network is of the same size as the input (*e.g.* in denoising), we need to expand the extracted feature maps back to the original size. We

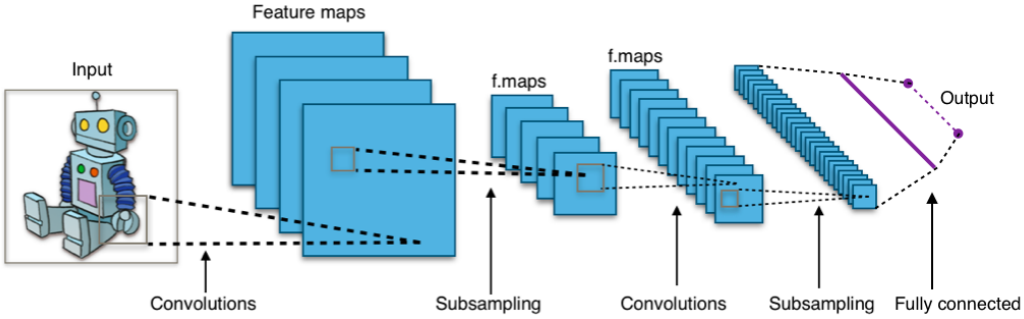

Figure 5. Example of CNN architecture (Aphex34, CC BY-SA 4.0).

discuss this process in the following section 2.2.4.

CNNs architecture designs can be quite deep, and thus more difficult to train than shallow networks. One way to improve the training for deeper networks is to use residual blocks famous from ResNet neural network architectures [12]. Residual block consists of few convolutional layers, where the output of the final convolution directly adds to the input of the block (Figure 6). This residual connection between input and the ouput of the block allows better information flow between the layers which can increase the performance of the networks.

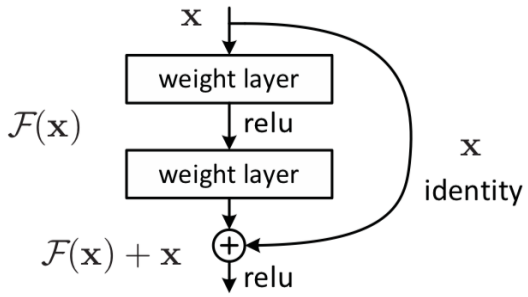

Figure 6. Example of residual block [12].

2.2.4 U-Net

U-Net is a widely used fully convolutional neural network architecture which was originally intended for biomedical image segmentation [13].

CNNs are often trained for classification tasks where the model's output is a single discrete label. However, there are cases where the output also needs to be an image (*e.g.*, semantic segmentation, denoising). In such instances, CNNs also have to expand after contracting. U-Net offers a solution by having an encoder and a decoder.

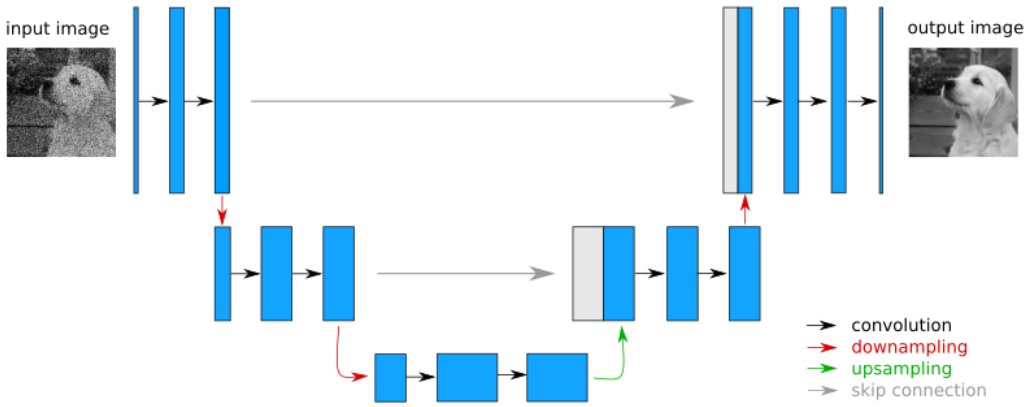

Figure 7. U-Net architecture for denoising. Rectangles show the shape change of the input tensor, width of the rectangles represents the number of channels and rectangle's height shows the size of the spatial dimensions. Name of the network comes from the shape.

Encoder, the first part of the U-Net, looks like a usual CNN: input image is convolved and downsampled in multiple steps, and the number of image channels is increased — the input image is transformed into high-level feature space. The decoder is symmetrical to the encoder: decoder output is upsampled, and the number of channels is decreased through the convolution until the input reaches its original dimensions. Features from the encoder path are concatenated with upsampling outputs via skip connections to bring localized information to the final output.

Encoder and decoder are divided into operational blocks bounded by downsampling and upsampling operations. These blocks are realised as residual blocks. The symmetry of encoder and decoder ensures that the final output has the same dimensions as the input, and thus U-Net can work with images with many different resolutions.

Denoising neural networks need an architecture that takes an image as an input and outputs an image. This requirement makes U-Net a good fit for such networks. An example of denoising U-Net is shown in Figure 7.

2.3 Denoising with neural networks

Denoising neural network aims to reconstruct a clean image from a noisy one. A classical way to train a denoising neural network would be to add artificial noise to a set of clean images and then use the pairs of clean and noisy images to train a CNN. This strategy is called *Noise2True*. In biological imaging, this approach includes networks such as Content-Aware Image Restoration (CARE) for fluorescence microscopy [5] and Deep-STORM for super-resolution single-molecule microscopy [14]. Sadly, this

method is not usually feasible: clean images often do not exist because it is prohibitively time-consuming to gather them in large numbers.

Lehtinen *et al*. propose a solution to this problem with *Noise2Noise* (N2N) - denoising method, which doesn't require clean data and can learn to denoise just from looking at noisy images [15]. To explain this method, let us consider a set of unreliable measurements $(x_1, x_2, \ldots)$ of some value. We can minimize a mean squared error (L_2 loss function) $L(\hat{y}, x) = (\hat{y} - x)^2$ to find an estimate $\hat{y}$ of the true value. Minimizing this loss gives us $\hat{y} = \mathbb{E}_x\{x\}$. Thus, the best estimate is the average of all noisy values. Now coming back to image denoising, if we train a neural network with pairs of noisy and clean images and use L_2 loss, the network learns to output the average of all the plausible explanations for the noisy images (one noisy image could be explained by multiple clean images). This results in blurry images, and it is usually an unwanted side-effect. However, this also means that the clean training data could be corrupted with zero-mean noise, and it wouldn't change what the network learns. This insight is the basis for *Noise2Noise*. The authors demonstrate it by training a network only with pairs of independently corrupted images, and it still learns to denoise.

The drawback of *Noise2Noise* is that it needs multiple noisy samples of the same frame to work. For real-world use cases, this requirement can be unattainable. This limitation is removed with the introduction of *Noise2Void*, which only needs a single noisy sample per image for training [6]. It operates on two assumptions: (i) signal is not pixel-wise independent, (ii) noise is conditionally pixel-wise independent given the signal. *Noise2Void* uses blindspot denoising: to denoise a pixel in an image, other pixels of the same image are used, but never the pixel itself. It is implemented by pixel masking: in training, a small percentage of pixels are masked, and their original noisy values are used as target values. The loss function is modified to zero for all the non-masked pixels. This blindspotting strategy is vital when the noisy image is used both as an input and target image. Otherwise, the network would be able to learn an identity function and output the same noisy image it received as an input. Similar blindspotting strategies are also used in *Noise2Self*

This blindspotting method is further improved with *Noise2Same*, which argues that the algorithm would be handicapped if it is blind to the pixel to be denoised and strict blindness is not needed [4]. A more complex strategy is used instead to avoid learning the identity function. For each image in training, two forward passes through the network are made (Figure 8). Once with an original noisy image and once with a noisy image where some pixels are masked. *Noise2Same* loss is a combination of the outputs of both forward passes and is given by

$$\mathcal{L} = \mathcal{L}_{rec}/m + \lambda_{inv}\mathcal{L}_{inv}, \tag{5}$$

where the first term is the reconstruction loss, second is invariance loss, m is the number of dimensions and λ_{inv} is a hyperparameter. The loss function has two terms:

(i) for reconstruction loss the output of the unmasked forward pass is compared to it's input with a mean squared error (MSE), (ii) for invariance loss the output of the masked forward pass is compared to the output of the unmasked pass for square root of invariance MSE. This second term of loss ensures that the network will not learn an identity function. Reconstruction and invariance loss equations are given by:

$$\mathcal{L}_{rec} = \mathbb{E}_x||f(\boldsymbol{x}) - \boldsymbol{x}||^2 \tag{6}$$

$$\mathcal{L}_{inv} = \mathbb{E}_J\left[\mathbb{E}_x||f(\boldsymbol{x})_J - f(\boldsymbol{x}_{J^c})_J||^2/|J|\right]^{1/2} \tag{7}$$

where J is a subset of m dimensions and represents the masked pixel space and J^c denotes the complement of J. Therefore $\boldsymbol{x}$ stands for the noisy image, $\boldsymbol{x}_{J^c}$ for the masked noisy image (all dimensions except J) and $f(\boldsymbol{x})_J$ denotes the values of $f(\boldsymbol{x})$ on J.

N2S is currently regarded as the state-of-the-art method for neural network based self-supervised denoising.

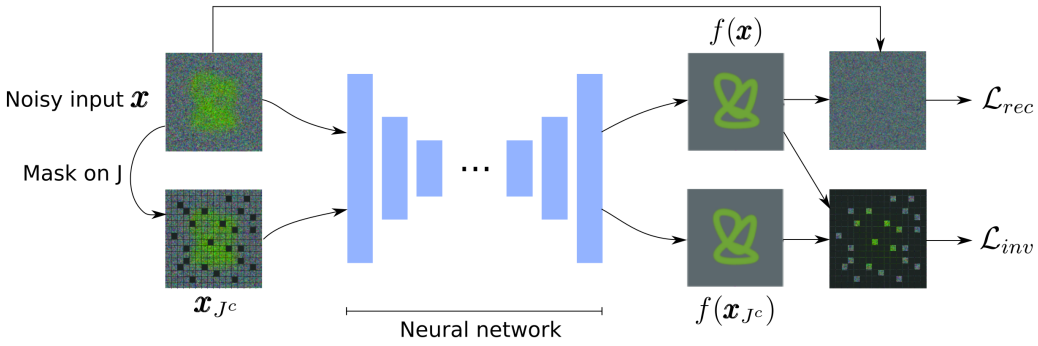

Figure 8. *Noise2Same* framework. $\mathcal{L}_{rec}$ is calculated over whole image while $\mathcal{L}_{inv}$ is only calculated on masked pixel positions.

2.4 Fast Fourier Convolutions

CNNs are not optimal for tasks that require combining spatially far apart information (*e.g.*, human pose estimation, image inpainting). Convolutional layers have limited receptive field (frequently 3×3 pixels), and so the network must be quite deep to bring spatially distant input points together. This can be a problem as deeper networks are usually harder and more expensive to train. *Fast Fourier convolution* (FFC) offers a solution when a non-local receptive field is desired [3].

Chi *et al*. introduced a new convolution block where some of the convolutions are carried out in the frequency domain thus obtaining a global receptive field. This FFC

block can be used to replace vanilla convolutions in conventional CNNs. Authors of FFC demonstrate elevated accuracies on vision benchmark datasets (ImageNet [16], MS-COCO human keypoint detection [17], Kinetics for video action recognition [18]) when switching convolutional layers to FFC in state-of-the-art CNNs. We will go over relevant Fourier theory, before introducing the design of the FFC.

2.4.1 Fourier theory

Fourier Transform (FT) is powerful tool in signal processing. It takes a function of time or space and transforms it into a function of temporal frequency or spatial frequency. In essence, FT deconstructs the signal into sinusoids it is composed of. General formula of FT for $f(t)$ is given by

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \omega} dt, \tag{8}$$

where ω is frequency and $F(\omega)$ represents, what frequencies are present in $f(t)$. We can restore the original signal from $F(\omega)$ with the inverse FT:

$$f(t) = \int_{-\infty}^{\infty} F(\omega) e^{2\pi i t \omega} d\omega \tag{9}$$

For a simple example of FT, we can consider sound. Sound waves are air molecules vibrating and, in turn, changing the air pressure, which allows us to perceive it. Sounds are usually a sum of multiple pure tones (single tone = air molecules vibrating with a certain frequency), and by listening, it is hard to tell what are the underlying tones that create that sound. If we apply FT to a sound signal (air pressure as a function of time), we get a function that characterizes the tones present.

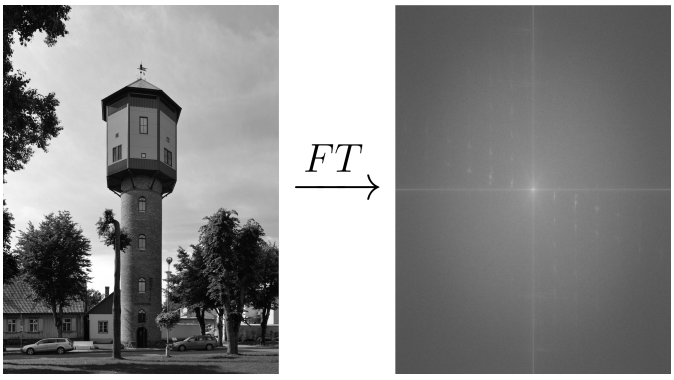

Figure 9. Example of 2D FT of natural image (Ivar Leidus, CC BY-SA 3.0).

Two-dimensional FT is usually applied to images. It also breaks the image down to the sinusoidal components, but in this case, the sinusoids serve as spatial frequencies

instead of temporal. 2D FT of an image can also be plotted as an image, but taking the modulus of the transform is needed as it is a matrix of complex numbers (Figure 9). Each pixel in such FT image represents a different sinusoidal component (also called a grating) of the original image. Pixel's value is the amplitude of the grating, the vertical distance from the middle of the image represents the orientation of the grating, and the horizontal distance represents the frequency of the grating (Figure 10). Information about the phase of the gratings is encoded in the complex Fourier matrix but not in the image itself (we show absolute values). Also, note that FT images are symmetrical to the center.

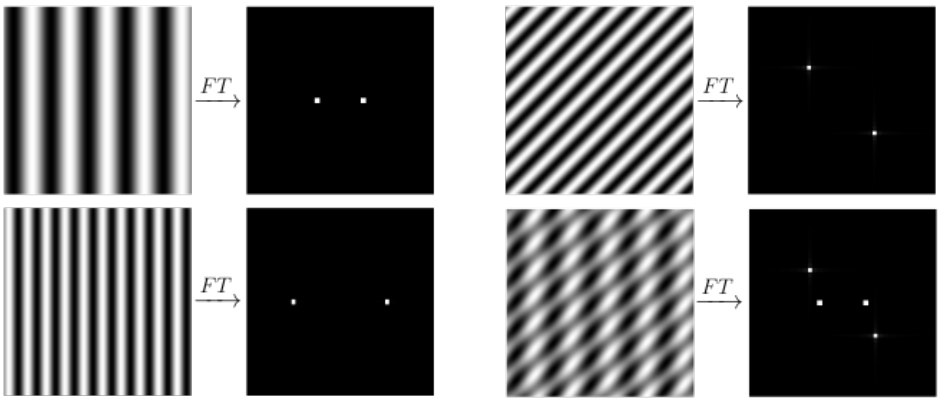

Figure 10. Examples of 2D FT of gratings to show the meanings behind the pixels of transformed images. FT images are zoomed in to the center. Bottom right image is a sum of two gratings from the top row.

In computer science, discrete Fourier transform is used instead of the continuous one as it is computationally less expensive and digital signals are already discrete. The general formula of DFT is defined by

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N}kn}. \tag{10}$$

DFT is usually implemented with a fast Fourier transform algorithm.

2.4.2 FFC Block Design

The key idea behind FFC comes from convolutional theorem in Fourier theory: convolution in the time/spatial domain is equivalent to a point-wise multiplication in the frequency domain. Thus, we can update the input image in the frequency domain, and it has a global effect on the spatial domain, giving us a non-local receptive field.

FFC is divided into two branches, which have different types of receptive fields: local and global. The local branch works as a normal convolutional layer. The global branch operates in the spectral domain and carries out global updates to the input. Input tensor to the FFC is split channel-wise between the two paths. Branches also exchange information with a local to global and global to local connection (Figure 11).

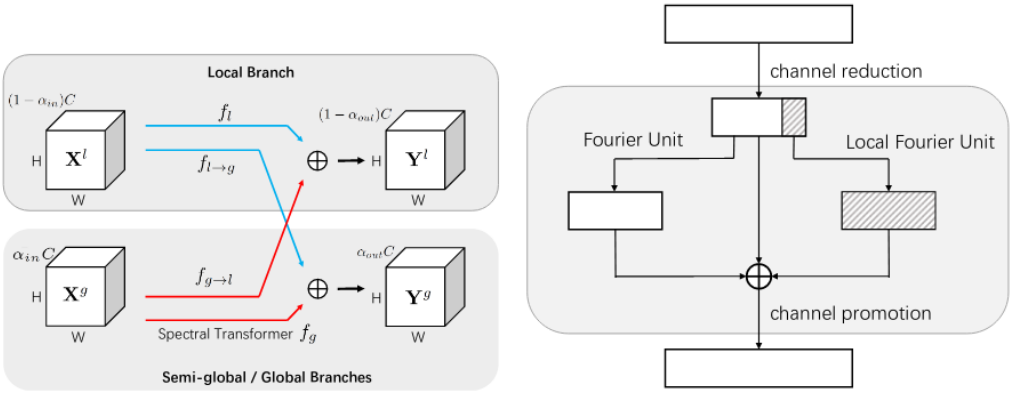

Figure 11. **Left**: Architecture of Fast Fourier Convolution. "$\bigoplus$" means element-wise sum. In this figure, $a_{in} = a_{out} = 0.5$. **Right**: Architecture of the spectral transformer. Quarter of the channels are also used for Local Fourier Unit.

Let $\mathbf{X}$ be the input to the FFC and let it have dimensions of $H \times W \times C$ (height, width, channels). At the start of FFC, the input is split along the channel dimension to local part $\mathbf{X}^l$ with $(1 - \alpha_{in})C$ channels and global part $\mathbf{X}^g$ with $\alpha_{in}C$ channels. α_{in} is the ratio of channels allocated to the global branch. The output of the FFC is also divided into two parts of $\mathbf{Y}^l$ and $\mathbf{Y}^g$ and parameter α_{out} controls the channel allocation between the outputs. Forward pass of the FFC can be then described by following formulas:

$$\mathbf{Y}^l = f_l(\mathbf{X}^l) + f_{g\to l}(\mathbf{X}^g), \tag{11}$$

$$\mathbf{Y}^g = f_g(\mathbf{X}^g) + f_{l\to g}(\mathbf{X}^l). \tag{12}$$

f_l acts as a branch with local perception field and is implemented by a normal convolutional layer. Likewise, $f_{g\to l}$ and $f_{l\to g}$ that act as information exchangers between the two branches are implemented with a convolutional layer. f_g has a global receptive field and is called spectral transformer.

Spectral transformer (Figure 11) begins with a 1×1 convolution that halves the number channels to reduce the computational costs. Next, the remaining channels are inputed to the Fourier Unit and Local Fourier Unit. Both carry out the same spectral convolution but in different boundaries. Outputs of the Fourier Unit, Local Fourier Unit,

and the input are summed element-wise and a 1×1 convolution restores the original amount of channels.

Fourier Unit conducts the FFT on the whole image at once, while Local Fourier Unit splits the image into four separate quarters first. Hence, Local Fourier Unit has a semi-global receptive field. The effect of Local Fourier Unit varies between the tasks and it can also be turned off. Authors of the original paper demonstrate that Local Fourier Unit clearly complements Fourier Unit in image classification task. Figure 13 explains the input handling in Local Fourier Unit.

```
def FU(x):
    # x: input features with shape [N,C,H,W]

    # y_r / y_i is the real / imaginary part of the results of FFT, respectively
    y_r, y_i = FFT(x) # y_r/y_i: [N,C,H,⌊W/2⌋+1]
    y = Concatenate([y_r, y_i], dim=1) # [N,C*2,H,⌊W/2⌋+1]
    y = ReLU(BN(Conv(y))) # [N,C*2,H,⌊W/2⌋+1]
    y_r, y_i = Split(y, dim=1) # y_r/y_i: [N,C,H,⌊W/2⌋+1]
    z = iFFT(y_r, y_i) # [N,C,H,W]

    return z
```

Figure 12. Pseudocode of Fourier Unit [3].

Pseudocode for Fourier Unit is shown in Figure 12. Fourier Unit starts with a FFT. The output of FFT is in two parts: real and imaginary. As the input to the FFT is real, the two parts are symmetric and we can ignore the imaginary output. To simplify computations, imaginary part is still used and concatenated together with the real part. Then a normal 1×1 convolution is conducted following batch normalization and ReLU activation. Note that larger kernel size for the convolution is not needed since any operation in the spectral domain has a global receptive field. Then the tensor is split again to real and imaginary part and inverse FFT is carried out to output a tensor with only real numbers.

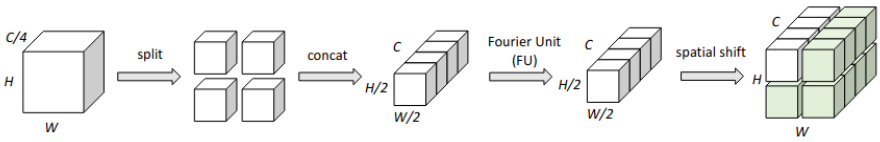

Figure 13. Input processing pipeline for Local Fourier Unit. Only quarter of channels are used for LFU. Tensor is split into four and then stacked together. After Fourier Unit the tensor is spatially shifted and replicated to achieve the original dimensions (light green features are copies of white ones) [3].

3 Methods

In this chapter, we discuss the setup details of the experimental work. First, we go over the neural network architectures tested. After that, we introduce the datasets used in experiments and strategies for data noising and augmentation. Lastly, we describe the evaluation metrics.

3.1 Base network architecture

All of our neural network designs use the vanilla U-Net as a base. We use different strategies to replace normal convolutions with FFC and investigate its effect on performance. The aim is to keep our own network implementations similar to the baseline in terms of the number of parameters and overall U-shape so that the results would be comparable. The baseline U-Net is the same as presented in *Noise2Same*. After the first convolution, the initial number of feature maps is 96; it is doubled with every downsampling operation. Architecture of the baseline U-Net is given in Table 1 and a visual representation in Figure 7. Details of normal residual blocks are shown in Table A1.

ID	Name	N_{out}
1	Input	n
2	Encoder Residual Block 1	96
3	Downsample	96
4	Encoder Residual Block 2	192
5	Downsample	192
6	Bottom Residual Block	384
7	Upsample	192
8	Concatenation (from 4)	384
9	Decoder Residual Block 2	192
10	Upsample	96
11	Concatenation (from 2)	192
12	Decoder Residual Block 1	n

Table 1. Overview of the baseline model. Outputs of encoder residual blocks get concatenated to the inputs of respective decoder blocks. Up- and downsampling are carried out with a 2×2 convolution, with a stride of 2 (in case of upsample the convolution is transposed).

3.2 FFC designs

FFC operates with a global and local path. The channel allocation is defined by α_{in} and α_{out}. As the shallowest layers are for low-level pattern recognition, it makes sense to set all of the channels to the local path at first ($\alpha_{in} = 0$). In intermediate layers we set $\alpha_{in} = \alpha_{out}$ and for the last consecutive FFC layer we converge all of the channels to local branch with $\alpha_{out} = 0$. It is important to note that the main novelty of the FFC is apparent when $\alpha_{in}, \alpha_{out} > 0$ as the spectral transformer applies only to the global-to-global connection. Still, we need the first and the last FFC for dividing channels between local and global branches and merging channels back to the local branch, respectively. We test two different designs for replacing normal convolutions with FFC: converging design and Twin Pass design.

3.2.1 Converging FFC designs

In the case of converging design, we isolate FFC branching only to a current encoder (or decoder) block level. This means that the input tensor for such block is only in a single local branch and the output is also in a single local branch. Global channels are self-contained inside a block. This design keeps the network architecture very modular as inputs and outputs of the block are identical to the vanilla U-Net. The downside of this design is the constant need to use FFCs for branch division and merging, which are not performing spectral convolutions.

Normally, U-Net residual block has two convolutions. This alone is not enough to implement such a converging FFC design. Convolution is needed for channel division and another for channel merging, leaving none for spectral convolution. However, if we use convolutions for downsampling and upsampling operations, we can switch them for FFCs and make this converging design work. For a single encoder block we have an FFC downsampling operation followed by two FFC convolutions (Figure 14). Problems arise if we want to mirror this design to a decoder. After FFC upsampling, there would be two branches present, but the input coming from the skip connection is only in the local branch. In that case, we use 1×1 convolution in the skip connection before concatenating channels together so that the model could better learn how to mix global and local channels. We further reference this approach as a vanilla FFC design. The details of this design are given in Table A2.

Another solution for the lack of convolutional layers is to introduce a new convolutional layer to the encoder and decoder blocks (Figure 15). That way, we can keep using regular convolutions for up and downsampling, and therefore we can utilize FFCs also in the decoder. This increases the number of trainable parameters, but it is straightforward to compare it to the vanilla U-Net with an added regular convolutional layer. We name this design an Extra Layer, and details of this design are given in Table A3.

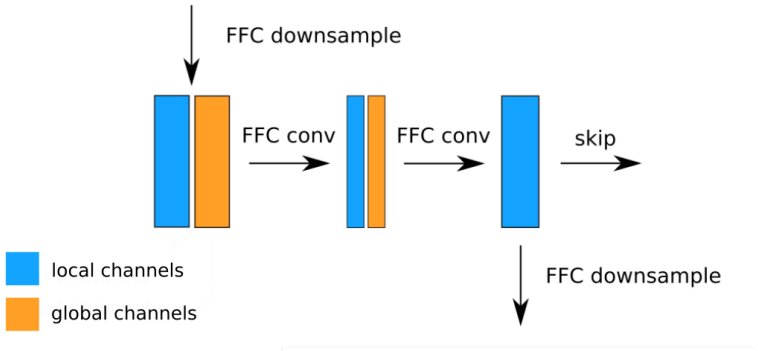

Figure 14. Vanilla FFC design. Modified residual block from U-Net encoder. Decoder design is mirrored. Residual block shortcut is not includded in the figure.

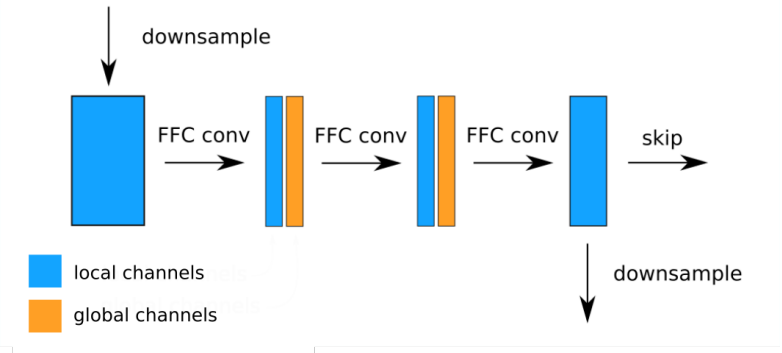

Figure 15. Extra Layer design. Modified residual block from U-Net encoder. Decoder design is mirrored. Residual block shortcut is not includded in the figure.

3.2.2 Twin Pass FFC design

In the Twin Pass design, local and global channels are not merged together at the end of every block (Figure 16). Both spatial and spectral information flow through the network in parallel. Both branches have a separate up- and downsampling layers and separate skip connections. Only the first and the last FFC layer deal with channel division and merging. This design is in line with the FFC implementation to the ResNet architectures by the authors of FFC. Details of Twin Pass are given in Table A4.

3.3 Datasets and data preparation

Below we describe datasets used in experiments and how the noise was generated for each dataset. The setup used is mostly based on setup used in *Noise2Same* paper [4]. For all datasets, we apply rotation and flipping data augmentations. We randomly pick 0.5%

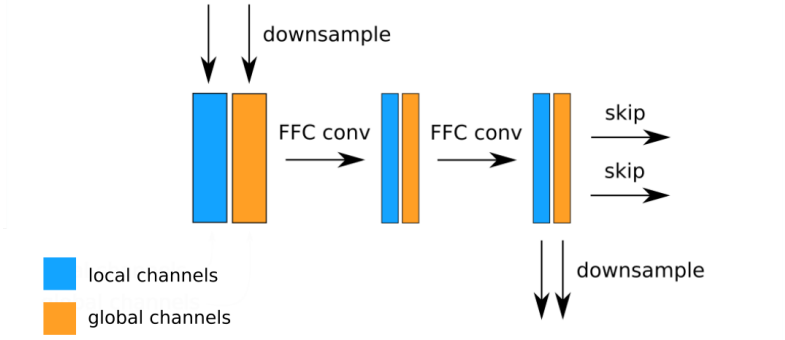

Figure 16. Twin Pass FFC design. Modified residual block from U-Net encoder. Decoder design is mirrored. Residual block shortcut is not includded in the figure.

of the pixels for image masking and replace them with Gaussian noise ($\sigma = 0.2$).

3.3.1 BSD68

BSD68 is dataset of grayscale natural images. It is part of The Berkeley Segmentation Dataset and Benchmark [19]. For training, patches of size 128×128 are cropped from each of the 400 images in range of $[0, 255]$. These images are corrupted with zero mean Gaussian noise ($\sigma = 25$). Trained models are evaluated on 68 images corrupted with the same type of noise.

3.3.2 Hànzì

Hànzì consists of 64×64 white on black images of hand-written chinese characters (pixel values within $[0, 1]$) [20]. The size of the dataset is 78174 images, which is split 90%/10% between training and testing. Images are treated with zero mean Gaussian noise ($\sigma = 0.7$) and Bernoulli noise (half pixels blacked out).

3.3.3 ImageNet

ImageNet is a large visual database of natural images for computer vision research [16]. For training we use the first 20 000 images from ImageNet ILSVRC2012 validation dataset (RGB values within $[0, 255]$). From these images 60 000 patches of size 64×64 are cropped for training. Another 1000 images are used for validation and testing respectively. Three types of noises are introduced to the images: zero mean Gaussian noise ($\sigma = 60$), Bernoulli noise ($p = 0.2$) and Poisson noise ($\lambda = 30$).

3.4 Evaluation

We evaluate the performance of models with Peak signal-to-noise-ratio (PSNR). It is calculated between a denoised image and the ground truth. Peak signal-to-noise ratio is given by

$$PSNR = 20 \cdot \log_{10} \left(\frac{MAX}{RMSE} \right), \tag{13}$$

where MAX is the maximum possible pixel value of the image and RMSE is the root mean squared error between the pair of images

$$RMSE = \sqrt{\frac{\sum_{i,j=1}^{m,n} (Y(i,j) - \hat{Y}(i,j))^2}{N}}, \tag{14}$$

where m and n are dimensions of the image and N is the number of pixels in image. Higher PSNR means that the images are more similar.

4 Results

All neural networks were implemented in PyTorch [21] and were trained on HPC cluster of University of Tartu on Tesla V100-PCIE-16GB or V100-SXM2-32GB GPUs. Our code is available at `https://github.com/JoonasAriva/noise2same.pytorch`. Training consisted of 80 000 iterations for BSD68 and 50 000 iterations for other datasets. We used batch size of 64 for each dataset.

We trained all three different designs (Vanilla, Extra Layer and Twin Pass) for each dataset mentioned. For each design we also trained a model with a vanilla decoder to investigate, whether FFC decoder justifies itself. For models, where skip connection would merge global channels to local or vice versa we added a 1×1 convolution as mentioned above. To address deviations from the *Noise2Same* design we also added different baseline models with extra layer in residual block or with 1×1 skip convolution. The ratio of global channels α was fixed to 0.5 for the FFC models if not stated otherwise. Besides the main experiments we also explored models with different values for α, or with Local Fourier Unit disabled.

Main results are given in Table 2 and denoising examples for each dataset are shown on Figures 17 to 19. On Hànzì most of the FFC models beat the baseline by a substantial margin. On BSD68 FFC models match baseline results or slightly exceed them. On ImageNet our model's performance slightly declines compared to the baseline. We compare our best performing models to other denosing models in Table 3. We see that on Hànzì our models beat current state-of-the art models including supervised models.

4.1 Local Fourier Unit experiments

The effect of Local Fourier Unit (LFU) is summarized in Table 4. For each dataset we picked the best performing designs and retrained them without LFU. Removing LFU decreases computational complexity but at the same time reduces denoising performance in most cases. Surprisingly, ImageNet denoising gets enhanced by removing LFU, but it still is overshadowed by the baseline.

4.2 Global ratio experiments

The authors of FFC set the ratio of global channels α to 0.5. While acknowledging that the importance of α could heavily depend on the distribution of images, we conducted a study on Hànzì dataset to see how does changing global ratio affect the model's performance. We picked two designs and retrained these on different values for α in range of $[0.25, 0.75]$. We also added the baselines ($\alpha = 0$) for comparison. The results are given in Figure 20.

ID	FFC encoder	FFC decoder	extra layer	twin pass	1x1 skip conv	ImageNet	Hanzi	BSD68	Params	Inference time (ms)
1						22.84	14.83	28.14	5.8M	3.68
2	✓					22.34	15.19	**28.16**	5.5M	10.04
3	✓	✓		✓		22.19	15.21	28.13	4.4M	17.38
4					✓	22.81	14.84	28.13	6.0M	5.35
5	✓	✓			✓	22.49	15.18	**28.16**	5.4M	14.20
6	✓			✓	✓	22.34	15.28	**28.16**	5.3M	11.46
7			✓			**23.06**	15.42	**28.16**	7.9M	6.39
8	✓	✓	✓			22.65	**15.72**	28.13	7.6M	13.67
9	✓		✓			22.39	15.04	28.09	7.6M	9.71
Noisy input						9.69	6.45	20.19	-	-

Table 2. Comparison among different denoising designs on different datasets, in terms of Peak Signal-to-noise Ratio (PSNR). Bold numbers indicate best performance for the dataset. Checkmarks indicate what components are present in the network. Different designs are ordered into three groups: designs which don't need 1×1 skip convolution, designs which need it and finally extra layer designs (also no need for 1×1 convolution). Baseline for each group is in top row. Parameter count and average inference time for a single 64×64 image is also given for networks.

	Methods	Datasets		
		ImageNet	HànZì	BSD68
Traditional	Input	9.69	6.45	20.19
	NLM [1]	18.04	8.41	22.73
	BM3D [2]	18.74	10.90	28.59
Supervised	Noise2True	23.39	15.66	29.06
	Noise2Noise [15]	23.27	14.30	28.86
Self-Supervised	Noise2Void [6]	21.36	13.72	27.71
	Noise2Self-Noise [22]	20.38	13.94	26.98
	Noise2Self-Donut [22]	8.62	13.29	**28.20**
	Noise2Same [4]	22.26	14.38	27.95
	Noise2Same (ours)	**22.84**	14.83	28.14
	Extra Layer (enc&dec)	22.65	**15.72**	28.13
	Twin Pass (enc)	22.34	15.28	28.16

Table 3. Comparisons among denoising methods on different datasets, in terms of Peak Signal-to-Noise Ratio (PSNR). Values for other models are taken from *Noise2Same* paper [4]. *Noise2Self-Noise* and *Noise2Self-Donut* refer to two masking strategies mentioned in [22], where the original results presented in [22] are produced by the noise masking. Bold numbers indicate the best performance among self-supervised methods.

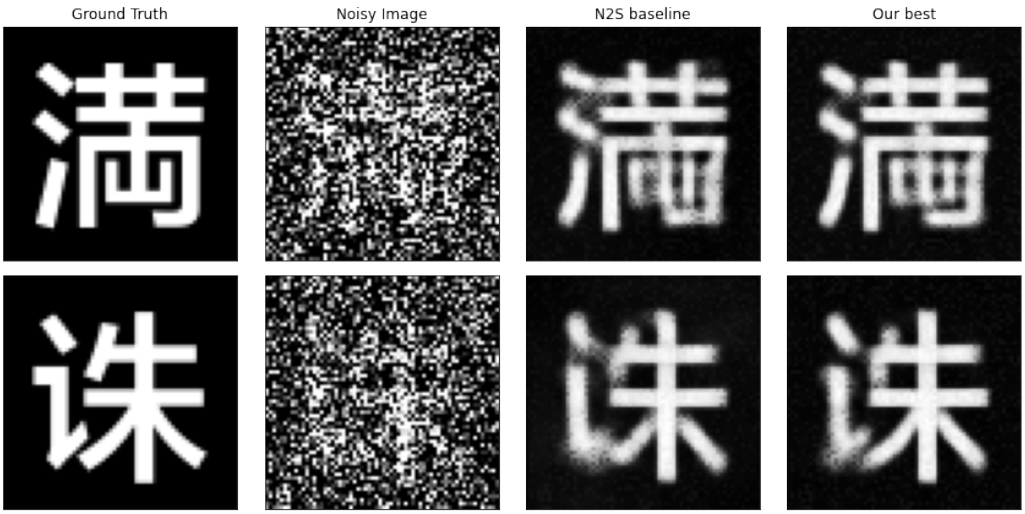

Figure 17. Visualization of testing results on Hànzì dataset. We compare our best model (extra layer design with FFC encoder and decoder) against vanilla *Noise2Same* baseline.

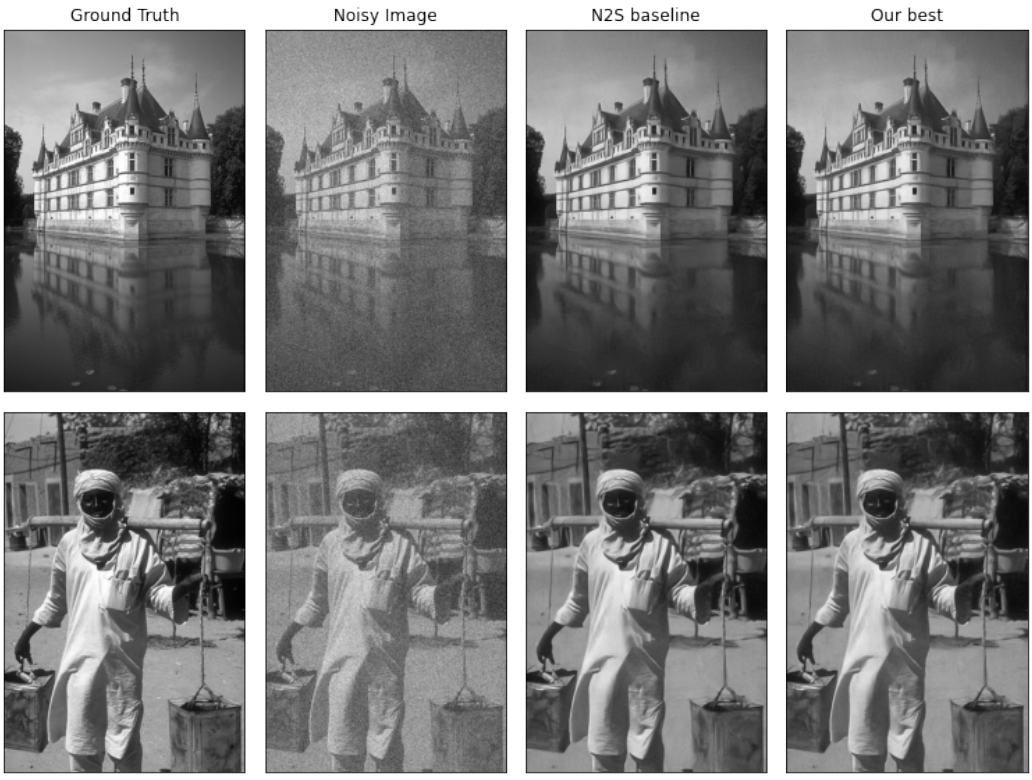

Figure 18. Visualization of testing results on BSD68 dataset. We compare one of our best models (twin pass without FFC decoder) against vanilla *Noise2Same* baseline.

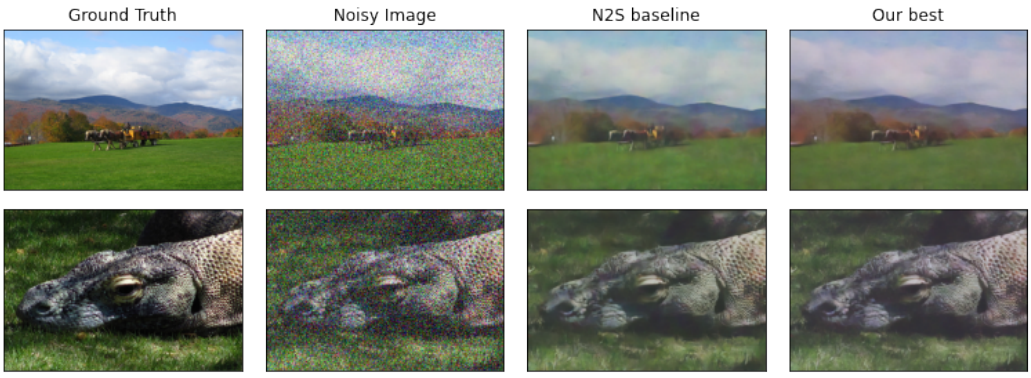

Figure 19. Visualization of testing results on Imagenet dataset. We compare our best model (extra layer design with FFC encoder and decoder) against vanilla *Noise2Same* baseline.

Dataset	Design	PSNR change	Params change
Hànzì	Extra Layer (encoder + decoder, 8)	15.72 → 15.68	-0.6M
Hànzì	Twin Pass (encoder, 6)	15.28 → 15.21	-0.6M
BSD68	Vanilla (encoder, 2)	28.16 → 28.15	-0.3M
ImageNet	Extra Layer (encoder + decoder, 8)	22.65 → 22.71	-0.6M

Table 4. Effect of disabling LFU in a model. We reference respective network designs by their IDs in Table 2.

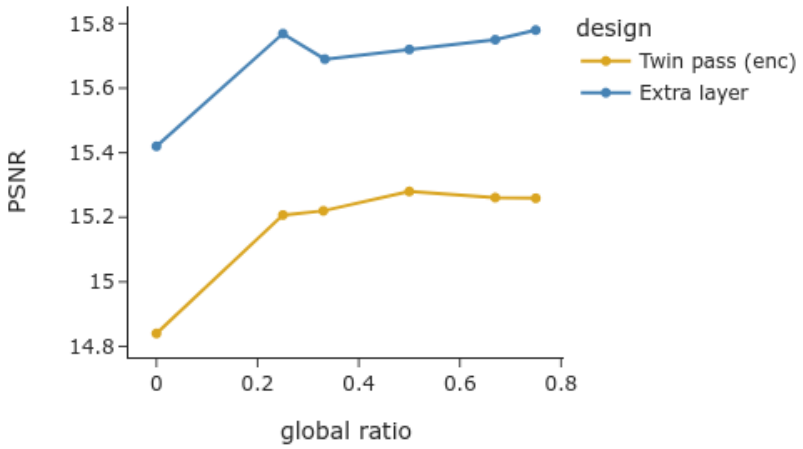

Figure 20. Performance change in relation to feature maps dedicated to spectral transformer in FFC. Used Twin Pass design has only FFC encoder.

5 Discussion

The effect of FFC on performance varies among datasets. We observe a slight decline in performance on ImageNet, modest gains on BSD68, and substantial improvements on Hànzì. The second effect of FFC is on parameter count. Models with FFCs have significantly fewer parameters than their baseline counterparts, and multiple times these models still match or exceed baseline results. The decrease in parameters is because in the frequency domain, there is no need for a larger kernel size than 1x1. Thirdly, models with FFCs are up to multiple times slower than those without (we measured average inference time for a single image). This could come from the lack of optimization of Fourier transforms for GPUs.

FFC brings the most significant improvement to Hànzì. All of the networks with FFCs beat their respective baselines on Hànzì dataset. Clean images from Hànzì can most likely be represented with few components in frequency space (straight uniform lines, sharp edges) and this might give an advantage to networks with FFC. Based on PSNR, the clear winner is the Extra Layer with FFC encoder and decoder (visualized in Figure 17). Visually comparing it to the baseline, we can see that in some cases, our model can handle sharp edges better and the characters are "diffusing" less to the background. Extra Layer design has more parameters than the baseline, but the parameter count is not a clear measure of the model's success. Twin Pass designs have up to 25% fewer parameters than the original baseline and still perform much better than that.

Natural image denoising (BSD68, ImageNet) did not gain as much from FFC as character denoising did. For BSD68 multiple models still beat the baseline, but only marginally. We have visualized the denoising results for the Twin Pass model without FFC decoder against the baseline (Figure 18). Visually, we didn't observe any differences between the baseline and the model. BSD68 was only corrupted with Gaussian noise, and it seems that for this case, the baseline *Noise2Same* has already solved the problem. Still, our models give numerically better results with fewer trainable parameters.

Vanilla *Noise2Same* beats our models on ImageNet dataset. We noticed that in some cases, our models tend to emphasise the red channel of the image while denoising (Figure 19). The image difference besides the shift to the red channel seems minimal. On ImageNet, the task of self-supervised denoising (with that sort of image corruption) is definitely not solved, as even the baseline images are duller in color and less sharp compared to the clean images. If we compare BSD68 (grayscale images) results to ImageNet (color images) results, we can hypothesise that models with FFCs have problems adapting to multichannel images.

The main experiment results do not indicate that there is a strictly superior FFC design for all of the cases. However, the two most promising designs seem to be Extra Layer with both FFC encoder and decoder and Twin Pass with only FFC encoder. Interestingly, Extra Layer benefits from FFC decoder, but Twin Pass does not.

Local Fourier Unit (LFU) study shows that most models lose slightly on PSNR when

LFU is disabled. Surprisingly ImageNet gains PSNR with this move. We suspect that this might be related to how color channel information is distributed throughout the input tensor and how channels for LFU are chosen. This needs further investigation.

The results of the global ratio study are mostly inconclusive. Changing global ratio affects performance, but the effect is quite subtle. For Extra Layer α values 0.25 and 0.75 both beat the default 0.5. On Twin Pass (only FFC encoder) the default $\alpha = 0.5$ is still the best. All of the FFC models are still on another level than the respective baselines in terms of PSNR. In the future, it is worth considering experimenting with very low and very high values of α, which were currently excluded as we believed that the optimal value is somewhere around 0.5.

5.1 Comparison to other models

Table 3 compares our best performing models (Twin Pass with FFC encoder and Extra Layer with FFC encoder and decoder) against other models. Curiously, our implementation of *Noise2Same* is better than the one presented in the paper. Difference might come from the reimplementation of *Noise2Same* in PyTorch as the original implementation was in TensorFlow [23]. Still in some cases our models manage to exceed the elevated baseline and in case of Hànzì we achieve state-of-the-art results over all methods including supervised. It is debatable whether it is a fair comparison though. Our Extra Layer baseline also performed very well compared to vanilla *Noise2Same* and so the improved performance does not solely come from the use of FFC, but also from the increased parameter count. Maybe other methods based on CNNs could also benefit from added layers? Nevertheless we saw that Extra Layer FFC design substantially improves on the Extra Layer Baseline.

5.2 Future work

We demonstrated that self-supervised denoising neural networks with FFC have potential and are worth investigating further. Still, there are many topics to tackle. Inspired by the improvement in Hànzì denoising, we would like to examine denoising on other image datasets that have contrasting features and well-defined structure (e.g number plates, biomedical images). We could also test how FFC affects denoising in the case of 3D data. Furthermore, we could try if it is possible to perform signal processing analysis on datasets to preemptively tell if it is worth using FFC for denoising or not. Also, we measured that denoising with FFC can be several times slower than without it. FFC further optimization for hardware could bring down the network training times and increase the denoising speed.

6 Conclusion

The aim of this thesis was to combine *Fast Fourier Convolutions* (FFC) with self-supervised neural networks for image denoising. FFCs have improved image classification CNNs and we investigated whether they could also improve denoising CNNs.

The theoretical part of the thesis is dedicated to reviewing principles of digital denoising, artificial neural networks and FFC. In the practical part, we designed and implemented multiple methods for replacing regular convolutions with FFC for the U-Net architecture (Vanilla FFC, Extra Layer, Twin Pass). We tested these modified U-Nets on two datasets consisting of natural images (BSD68 and ImageNet) and one consisting of Chinese characters (Hànzì). Datasets were corrupted with different types of noise and the models were trained using only noisy images.

We compared our models with FFCs to baseline models and observed slight improvements on BSD68, some decline on ImageNet and state-of-the-art results for Hànzì. Among our model designs, the best performing were Extra Layer (with FFC encoder and decoder) and Twin Pass (with just FFC encoder), yet no design was superior in all experiments.

Experimental results confirm that FFC can improve denoising performance in some cases, and this topic is worth exploring further.

7 Acknowledgements

The author acknowledges the support from the Institute of Computer Science at the University of Tartu during the writing of this thesis. Also, I am grateful to my supervisor, Mikhail Papkov, for his patience and all the support and advice he gave me.

References

[1] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011.

[2] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[3] Lu Chi, Borui Jiang, and Yadong Mu. Fast fourier convolution. *Advances in Neural Information Processing Systems*, 33:4479–4488, 2020.

[4] Yaochen Xie, Zhengyang Wang, and Shuiwang Ji. Noise2same: Optimizing a self-supervised bound for image denoising. *Advances in Neural Information Processing Systems*, 33:20320–20330, 2020.

[5] Martin Weigert, Uwe Schmidt, Tobias Boothe, Andreas Müller, Alexandr Dibrov, Akanksha Jain, Benjamin Wilhelm, Deborah Schmidt, Coleman Broaddus, Siân Culley, et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nature methods*, 15(12):1090–1097, 2018.

[6] Alexander Krull, Tim-Oliver Buchholz, and Florian Jug. Noise2void-learning denoising from single noisy images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2019.

[7] Ibm. `https://www.ibm.com/cloud/learn/neural-networks`.

[8] 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.

[9] 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.

[10] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] 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.

[14] Elias Nehme, Lucien E Weiss, Tomer Michaeli, and Yoav Shechtman. Deep-storm: super-resolution single-molecule microscopy by deep learning. *Optica*, 5(4):458–464, 2018.

[15] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. *arXiv preprint arXiv:1803.04189*, 2018.

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755. Springer International Publishing, 2014.

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[19] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[20] D.-H. Wang Q.-F. Wang C.-L. Liu, F. Yin. Casia online and offline chinese handwriting databases, proc. 11th international conference on document analysis and recognition (icdar), 2011.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[22] Joshua Batson and Loic Royer. Noise2self: Blind denoising by self-supervision. In *International Conference on Machine Learning*, pages 524–533. PMLR, 2019.

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

Appendix

I. FFC implementation details

Here we give a detailed overview of all the residual blocks and their implementation details. Baseline residual block is given in Table A1. Baseline block can be switched for one of the FFC residual blocks described in Tables A2 to A4. In all residual blocks (including baseline) there is a layer of batch normalization followed by ReLU activation before FFC or convolutional layer. The tables are given for encoder blocks, but decoder blocks are very similar. In case of decoder blocks the number of channels halves after the first FFC (or regular convolution) instead of doubling and downsample is replaced with upsample.

When replacing regular blocks with FFC blocks, downsample, upsample or concatenation layers need also to be modified in some (Tables A2 and A4). Whenever we replaced encoder or decoder blocks for FFC variants we also switched the bottom residual block. In tables N_{out} denotes the number of output feature maps for each layer and α_{out} signifies the ratio of feature maps in the global branch. All of the regular convolutions inside FFC are 3×3 if not stated otherwise. The spectral convolution in global-to-global connection is always 1×1.

ID	Name	N_{in}	N_{out}
1	Downsample Convolution (2×2)		n
2	Convolution (3×3)	n	$2n$
3	Convolution (3×3)	$2n$	$2n$
4	+ Shortcut Convolution (1×1)	n (from 1)	$2n$

Table A1. Overview of the baseline Encoder Residual Block. Input gets added to the output via shortcut convolution. We cannot use identity function for shortcut because input and output have different number of feature maps.

ID	Name	N_{in}	N_{out}	α_{out}
1	Downsample FFC (2×2)		n	α
2	FFC 1 (3×3)	n	$2n$	α
3	FFC 2 (3×3)	$2n$	$2n$	0
4	+ Shortcut FFC (1×1)	n (from 1)	$2n$	0

Table A2. Overview of the vanilla FFC encoder block. When imputing this block into a model we also have to change the downsample blocks before residual blocks (upsample blocks for decoder). Regular convolutions inside up- and downsample FFC act as up- and downsample convolutions described in Table 1. There is no need for up- and downsample spectral transformer as there are no global channels before sampling. Shortcut connection is also FFC as the output from downsample layer is in two branches. If both encoder and decoder are with this design, 1×1 convolution is added to the concatenation step, because encoder output is only in a single branch, while the decoder input (from FFC upsample) is in both branches.

ID	Name	N_{in}	N_{out}	α_{out}
1	Downsample Convolution (2×2)		n	0
2	FFC 1 (3×3)	n	$2n$	α
3	FFC 2 (3×3)	$2n$	$2n$	α
4	FFC 3 (3×3)	$2n$	$2n$	0
5	+ Shortcut Convolution (1×1)	n (from 1)	$2n$	0

Table A3. Overview of Extra Layer Encoder Residual Block. Input gets added to the output via shortcut convolution.

ID	Name	N_{in}	N_{out}	α_{out}
1	Downsample Convolution (2×2) $\times 2$		n in total	α
2	FFC 1 (3×3)	n	$2n$	α
3	FFC 2 (3×3)	$2n$	$2n$	α
4	+ Shortcut Convolution (1×1) $\times 2$	n (from 1)	$2n$ in total	α

Table A4. Overview of Twin Pass Encoder Residual Block. While using this block we also have to make changes to the up- and downsampling layers in the baseline U-Net (Table 1). In Twin Pass global and local branches do not converge before the end of the last Twin Pass residual block (where the final FFC has $a_{out} = 0$). Therefore we have two downsample and shortcut convolutions, one for each branch. The output of the block is in two branches and so concatenation step happens differently than in baseline U-Net. If both encoder and decoder are with Twin Pass, then the global and local branches get concatenated to respective inputs of the decoder block. If encoder is with Twin Pass and decoder is not then global and local channels are stacked together and convolution (1×1) is used before concatenation.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Joonas Ariva**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

   1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

   1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

   of my thesis

   Fast Fourier Convolutions in Self-Supervised Neural Networks for Image Denoising

   supervised by Mikhail Papkov

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, May 17, 2022