Signal Processing Transformations in Scene Recognition from Satellite Imagery

David J Florez
CS 230
sirdavid@stanford.edu

Toby Frager
CS 231N
tobiasfr@stanford.edu

Abstract

We propose that traditional signal processing transformations, namely the Fourier Transform and Wavelet Transform, extract meaningful information from visual data for the purpose of image recognition. We compare convolutional neural network performance on raw visual data to performance on data including channels of transformed data. We apply this technique to the task of recognizing different types of forest land as in the ForestNet paper published by the Stanford ML group. We compare training time, rate of convergence, and accuracy between data types over multiple models. We find that using the Fourier transform can improve performance drastically for simple models and speeds up training time for complex models. For the Wavelet transform, we find that there are moderate improvements on simple models, but there is only a slight decrease in performance for complex models.

1. Introduction

The challenge of image recognition rests on extracting important spatial features from rich, high-dimensional visual data. The field of signal processing offers mathematical tools which extract frequency information from spatial data in the form of integral transforms. We propose that the information in the transformed images may be more compact and more easily processed by neural networks, leading to faster training, higher data efficiency, or higher performance. We find that signal processing transformations robustly improve the performance of simple CNN models to the level of being comparable to deeper models, but that directly training deep models on transformed data hurts performance.

1.1. Related Work

Multiple papers indicate that the use of frequency domain transforms can improve the performance of machine learning models. [1],[5] use wavelet transformations to make classification tasks on complex data simple. Further, [4], [6] suggest that the Fourier transform leads to improvements in data compactness and model performance. We will use the architecture of [6] as a baseline later in the project.

ForestNet [3] is a project out of the Stanford ML group which applies CNNs to classify the different drivers of Forest Loss. We use their dataset and task in our investigation. While they use a vairety of additional indicators and scene segmentation to achieve high accuracy, we focus on using only the visual data.

Xu, et al[6] demonstrates that a CNN operating on frequency domain data (They use the discrete cosine transformation rather than Fourier) performs comparably and improves upon CNN's operating on spatial data. They employ complex data augmentation tools to achieve high perfomance on the ImageNet classification task. Mehrabkhani et al [4] also shows a similar result. We simplify from these approaches by treating only the data in a model-agnostic way to isolate the effects of the transformation.

Qin [5] and Anguita [1] demonstrate that the wavelet transform can usefully featurize data for prediction tasks. These papers demonstrate that wavelet transforms can effectively replace multiple neural network layers. We test if this means they can be integrated into deep models and provide an increase in performance.

1.2. Problem Statement

We take on the problem of classifying drivers of forest loss from satellite imagery. Our models take as input image data of a forest scene from a bird's eye view and classifies it into one of 12 different types, including "Logging", "Mining", and "Secondary Forest". We use overall categorical accuracy scores as our primary metric to judge model accuracy.

This problem is challenging as forests span immense amounts of land and it can be hard to make out even with a human eye what sort of disruption there is in a scene, as there is relatively low detail. This task is important as forest health is a critical part of the ecosystem and monitoring it quickly and accurately is necessary to have an active pic-

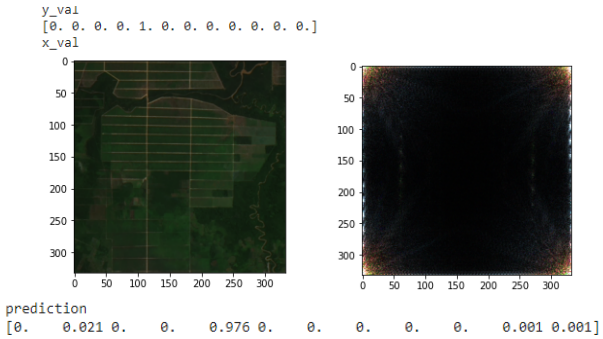

Figure 1. A sample image, its Fourier Transform and a prediction from Simple CNN trained on FT

ture.

1.3. Dataset

We are using the dataset from the ForestNet project from the Stanford ML group [3]. It consists of 2,756 satellite images of forest loss events in Indonesia taken between 2013 and 2016 with expert annotations of the driver of forest loss. The images come from the Tier 1 Landsat 8 satellite with 15m per pixel resolution. Each image is 332x332x3 of visible spectrum RGB values as well as 332x332x3 of infrared imagery. At this stage, we are restricting ourselves to the visible spectrum data. Each image is a composite of images of the same scene edited together to avoid cloud cover. Each image comes with polygon data outlining the boundary of the forest loss as well as metadata describing the altitude, and proximity to cities of the scene as well as metadata about the image specifications. We are not currently using any of the metadata in our investigation.

The only preprocessing or augmentation of the data was our experimental setup. We used the raw image data and transformed it with the Fourier Transform and the horizontal Haar wavelet transforms to run our experiments. We fed these directly into the models as well as concatenating them to the original data.

2. Methods

We processed the data into Raw, Fourier Transformed (FT), and Wavelet Transformed (WT) data. For the FT data, we take the magnitude of the complex values and normalize to [0, 255] so it has comparable values to visual data. We then trained various models on the Raw, FT, WT, and combination of Raw, FT, and WT datasets. We trained a 3-layer convolutional network and ResNet50V2 on each dataset.

2.1. Technical Approach

We transform our data through the 2D Fourier Transform and the 2D wavelet transform implemented in Numpy and Pywt. Our method will use **the same model the transformed and untransformed data**, and our results will be the **comparative performance** between these.

In theory, a deep convolutional network should be able to learn the Fourier transform or the wavelet transform approximately for the purposes of the task. We hypothesize that simple models will see improvement from the transforms because their limited expressiveness will be augmented by the transformation of the features, while complex models with high expressiveness will gain little or possibly decline in performance due to getting non-spatial data.

2.2. Integral Transforms

Signal processing makes use of various **integral transforms** to compress spatial information. For a given function $f(x)$ in the spatial domain. An integral transform T of the function can be described

$$(Tf)(u) = \int_{t_1}^{t_2} f(t)K(t,u)dt \tag{1}$$

For a given **kernel function** $K(t,u)$, this operation produces a function Tf over a new domain which captures different information about the function f depending on the kernel function. In signal processing, data is often treated as this function f over a spatial domain.

2.2.1 Fourier Transform

The **Fourier Transform** takes spatial data and outputs information about its gloabal frequencies. The strength of a frequency in a specific direction is recorded at the point in that direction in frequency space at distance from the origin corresponding to frequency. It is calculated

$$(\mathcal{F}f(\xi)) = \int_{\mathbb{R}^2} f(x)e^{-2\pi i x\cdot\xi}dx \tag{2}$$

This is also computed over discrete domains such as pixel data, returning data of the same size as the inputs. In practice, it is executed by an algorithm called the Fast Fourier Transform (FFT) which runs in $O(N\log(N))$ time complexity.

The transformed image, as in figure 1, has noise in the corners corresponding to the low scale texture of the image. Components in the center of the image correspond to the larger scale structural information. The repeating rectangles in the raw image of figure 1 manifest as a single rectangle in the center of the transformed image. We observe the Fourier transform compacting large scale structure into single local

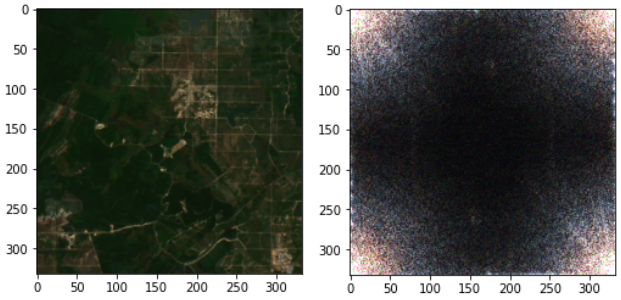

Figure 2. Raw image data and the magnitude of its Fourier transform

features. Additionally, the small-scale texture of the forest is captured in the corners of the image, so they take up only a small part of the transformed image.

This means that it is a good fixed way of obtaining information about structures in an image, which can correspond to semantic properties. In this case they are the spread out physical structures which the model aims to detect, i.e. plantations and mines. Additionally, the textural, non-semantic information that is shared between all examples, here the texture of the forest, is confined to a small part of the image, so it does not obscure or compete with the semantic information in the image.

This will lead to easier training, as models will not have to learn how to detect structural features. It also suggests that for simple models, which have limited expressiveness, the Fourier transform can improve performance, as it will have access to more direct semantic data contained in the larger image structures, so less expressiveness is required for accuracy. For deeper models, it is possible that the model can learn the Fourier transform or an even more useful representation, so it is not as clear whether this augmentation will help.

2.2.2 Wavelet Transform

The **wavelet transform** takes in spatial data and represents local frequency information. It acts with a kernel basis which is both oscillatory to capture frequency information and square-integrable, so it captures local properties. It is calculated:

$$[W_\psi f](a,b) = \frac{1}{\sqrt{a}} \int_{\mathbb{R}^2} \overline{\psi\left(\frac{x-b}{a}\right)} f(x)dx \qquad (3)$$

For a wavelet function ψ, the choice of which affects the output of the transformation. We use the Daubechies and Haar family of wavelet functions.

These wavelets extract different features from the data according to the character of the wavelet. As is visible, the Haar wavelet transformation highlights borders in the the

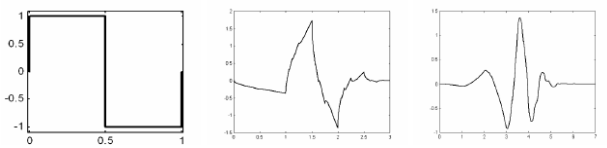

Figure 3. Examples of standard wavelets usd in the wavelet transform. Haar, Daubechies 4 and Daubechies 6 wavelets

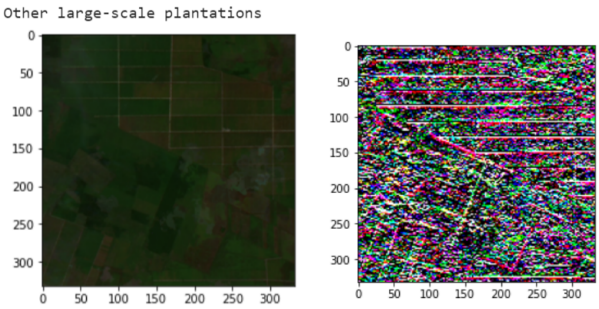

Figure 4. Raw image along with its Haar wavelet transform using horizontal coefficients

images, which emphasizes the useful features in the data. We expect that the transformed data will directly carry more semantic information, leading to improved performance for our simple model.

2.3. Models

We train on the task for two model architectures: Simple CNN and ResNet50V2. We choose these models as a scale of model complexity, measured by depth. Models were implemented with Keras using the Sequential API and the built-in ResNet.

2.3.1 Simple CNN

The architecture of Simple CNN is

- 3 layers of `3x3 conv + 2x2 maxPool`
- 1 layer of `1x1 conv`
- 2 layers of `FC with 128 hidden units`

This simple CNN is a benchmark where we expect to see improvements from the transformed data, and we can compare these improvements to the changes in performance for the more complex models.

2.4. ResNet V2

ResNet V2 [2] is a standard high-performing convolutional neural network that we use as a benchmark for complex models. We expect to see relatively little effect from using transformed vs raw data. For the augmented data, we implement ResNet with variable input channel number,

Data	Model	Val Acc	Val Loss
Raw	Simple CNN	.2939	3.4767
FT	Simple CNN	.4609	1.6493
WT	Simple CNN	.2791	3.2864
Raw	ResNet	.4651	2.7595
FT	ResNet	.3679	2.5365
WT	ResNet	.4100	1.900

Data	Model	Test Acc	Test Loss
Raw	Simple CNN	.2695	3.1157
FT	Simple CNN	.4027	1.8966
WT	Simple CNN	.3159	2.3895
Raw	ResNet	.4237	1.9842
FT	ResNet	.3174	2.7340
WT	ResNet	.3892	1.9437

Table 1. Accuracy and loss for combinations of models and data augmentations. Raw: Spatial data, FT: Fourier Transform, WT: Wavelet Transform. ResNet + WT figures approximate

similar to the method in [6], but we include the first convolution.

2.5. Experimental Procedure

We will benchmark each model by its performance on the untransformed data. We will compare the performance and training time between **raw data** and the **transformed data**.

For each combination of model and data, we train the network over short epoch numbers (10-20) to perform hyperparameter search, and select the best test set performance we find among hyperparameter settings trained over 100 epochs, keeping the model with the highest validation accuracy. We also keep track of the time per epoch for each experiment, as well as the rate of convergence. Models are optimized using the Keras Adam optimizer.

3. Results

We report validation and test set performance for each of our datasets and models. We also report the training time per epoch for each combination, and include some representative loss charts.

3.1. Simple CNN

With the simple model, preprocessing the data with the Fourier Transform produces an immense improvement in performance. During hyperparameter tuning, with only 20 epochs compared to 50 epochs for the raw data, the transformed data improved accuracy by 9% on validation and

Data	Model	Clock Time per Epoch
Raw	Simple CNN	1.5
FT	Simple CNN	2.8
WT	Simple CNN	3.0
Raw	ResNet	16.6
FT	ResNet	17.6
WT	ResNet	17.0

Table 2. Time in seconds to run a single epoch using Keras optimization

Data transformation	Transformation Time
FT	81.14
WT	21.21

Table 3. Time in seconds to transform complete dataset of 2757x332x332x3 visual data

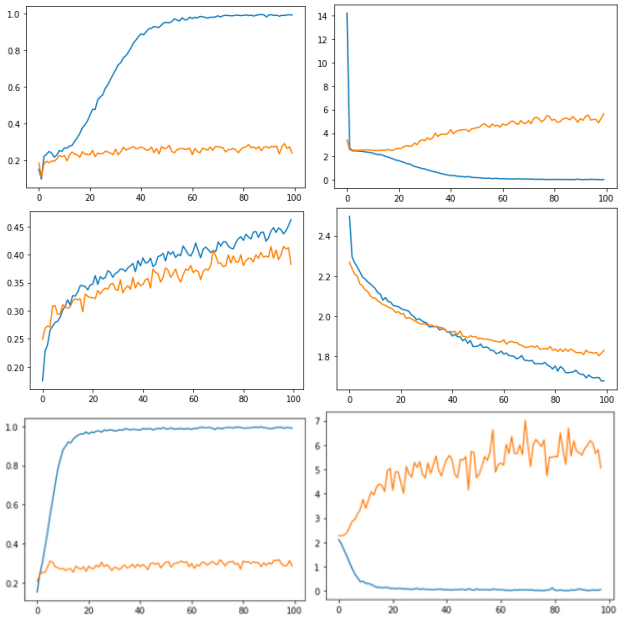

Figure 5. Validation(orange) and training(blue) accuracy (left) and loss (right) curves for Simple CNN over training epochs. Above is on raw data, middle is on FT data, and bottom is WT data

12% on test data. In the final result, we see a similar 7% and 13% boost, respectively.

Further, with Fourier transformed data, the model was much more robust to hyperparameter settings. While the model barely improved validation accuracy at all on raw data for most hyperparameter settings other than the optimal settings, the model on Fourier data trained to comparable accuracies with learning rates from 1e-7 to 1e-3 and dropout rates from 0 to 0.15.

We observe that with FT data, the model has much more expressive power. In this run, it was still learning at 100 epochs, but in others, it converged to about 40% accuracy

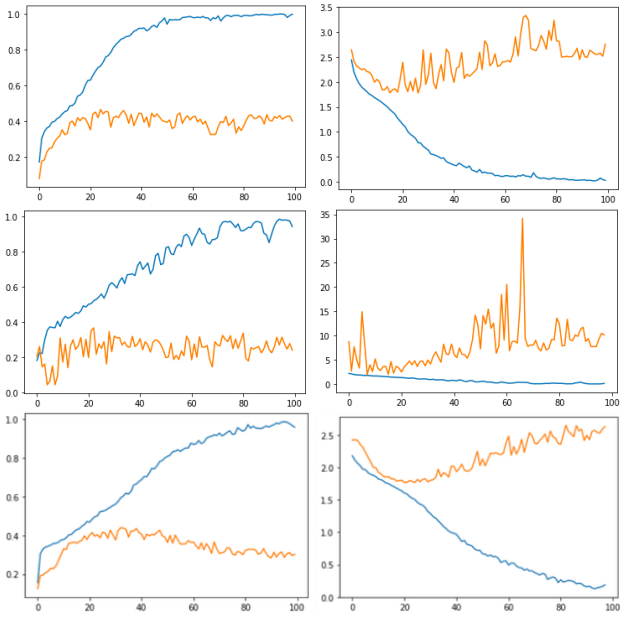

Figure 6. Validation(orange) and training(blue) accuracy (left) and loss (right) curves for ResNet over training epochs. Above is on raw data, middle is on FT, and bottom is on WT. First 2 epochs of loss omitted from loss on FT because val loss was so high it ruined perspective.

well before then. Note also that validation accuracy is not far below training accuracy, so the model really is learning more powerful semantics from the transformed image.

With the WT data, we see a modest improvement of 5% on test data on only 50 epochs. We see as well that Simple CNN trained on WT data is very robust to hyperparameter changes, as it trains to comparable accuracies from learning rates from 5e-3 to 5e=5 and dropout rates from 0.05 to 0.2.

3.2. ResNet

With raw data, we see that ResNet outperforms any data source and Simple CNN, although this improvement is not dramatic at 2% accuracy over Simple CNN and FT. We see that with FT data, however, ResNet performs considerably worse than the Simple CNN. ResNet + FT performs 10% worse on validation data and 11% worse on the test set. We see that ResNet + WT only performs about 2% worse, potentially within the variability of model performance.

The raw data, FT, and WT data proved moderately sensitive to hyperparameter settings. They learned outside of the optimal parameters but performance dropped off fairly steeply, to the tune of about 4% per factor of 10 in lr.

We see that ResNet had very unstable training on the FT data compared to the raw data. Though both models had peak performance at about 20 epochs, for raw image data, the accuracy and loss were fairly stable beyond that and while loss did increase afterwards, it was not drastic. On the other hand, both training and validation loss for the FT data were erratic, and validation loss increased a great deal after peak performance. Additionally, using the FT data, ResNet takes 80 epochs to reach approximately 100% training accuracy, compared to about 50 for raw data.

Model	Data	Test Acc	Clock Time per Epoch
ResNet	Gray + WT + FT	0.3832	10s

Table 4. ResNet run on augmented data featuring raw image, FT and WT of grayscaled image

We see that ResNet suffers both in its ability to fit data and its ability to learn generalizable information in the transformation from raw image data to FT data.

On the other hand, the WT data does not seem to impact the training of ResNet much at all. The loss and accuracy appear more stable and follows the same trajectory in both training and validation. The difference in performance is well within reasonable expectations for the variation of accuracy for ResNet on this task, so the decrease in performance is minor.

3.3. Augmented data

This experiment trained a ResNet model with one channel the raw grayscale image, one channel the Fourier transform of that image, and one channel the wavelet transform of that image. We see in a limited experiment that combining all three data types together even after removing color performs very well compared to the raw data. There is only a 4% dropoff in accuracy and a 40% increase in training speed. Though we have no baselines, this is an impressive figure suggesting that the combination of all three of these techniques is effective, making up for the complete loss of color information.

4. Conclusion

4.1. Key Results

- The performance difference between Simple CNN with FT and ResNet on raw data is small, and over many epochs, the computation time to train and run Simple CNN with FT is much faster.

- The performance of Simple CNN on WT data improves over raw data, and the performance of ResNet on WT data is comparable to its performance on Raw data.

4.1.1 Relative performance of transformed data

We observe that FT data leads to significant improvements to our simple CNN, both achieving higher performance on

test data and demonstrating a smaller gap between training and validation performance. Likewise, we see that the WT data leads to an improvement in model performance and a smaller gap between training and validation accuracy. The smaller gaps imply that the increase in performance is not coming from the model improving its ability to fit the training data, but from the training data representing more generalizable information than the raw image. This suggests that for the smaller network, for which expressiveness is a limiting factor, the transforms surface important semantic information in the image, which allows it to make more meaningful predictions.

Using ResNet, we observe that the FT data leads to significantly worse performance than using the raw data. We see a significant dropoff in both accuracy and the rate at which the model can fit the data. ResNet has a much harder time fitting FT data than raw data in general, suggesting that the ResNet architecture favors spatial domain data. This matches with the idea that deep convolutions, which have large receptive fields, do not capture information from FT data easily. In the frequency domain, large regions do not correspond simply to any single piece of information in the image, and locality, captured more with deep convolutional networks, is not as important.

Further, we see that even after the model fits the training FT data, it does not improve validation performance beyond 30%, suggesting that the FT data does not retain the semantic information effectively captured by ResNet.

With WT data, we observe moderate improvement to Simple CNN, achieving increased accuracy and more stable training, with higher resilience to hyperparameter changes. We also notice that the model is able to fit the training data much faster. This suggests that while the WT data does surface some semantic information, leading to higher test accuracy, it is doing much of its work by removing noise and irrelevant information from the data, allowing the model to fit to training data much more easily.

Using WT data for ResNet, we find that the accuracy is slightly reduced, but that patterns of loss and accuracy in training are very similar to raw data. Since the wavelet transform is a local operation, which does preserve spatial information, it is reasonable that it acts much like a few convolutional filters would, and so its effect on ResNet, already a very deep model, is minimal.

4.1.2 Efficiency of training

With Simple CNN, we notice an 87% increase in computation time per epoch on FT data and a 100% increase for WT data. Compared to raw data, and on ResNet, we see a 6% increase in computation time. The FT takes an additional approximately .029 seconds per example to process. Even including processing time for FT, Simple CNN with FT trains 5x faster than ResNet over 100 epochs and achieves similar results. WT data was very fast to create, at less than 0.01 seconds per example, and on ResNet it was only 2.4% slower. A more effective inclusion of WT data would likely not make training or prediction significantly slower.

4.1.3 Hyperparameter Results

With SimpleCNN, we observe an immense increase in stability with respect to hyperparameters switching from raw to FT data. With ResNet, the model performance had similar sensitivity to hyperparameters training on raw and FT data. Similarly, the WT data led to more robustness to hyperparameter change than did the raw data.

While further research would be needed to make a confident claim, we hypothesize that this is is also due to the transforms surfacing semantic information. This would make it easier for any model to distinguish between signal and noise and possibly lead to a nicer loss landscape, so it requires less balancing of hyperparameters to avoid overfitting.

4.1.4 Performance of Augmented Data

The fact that the Augmented Data performed comparably to the raw and WT data and better than FT data on ResNet despite the loss of color information suggests that these transformations add to the power of CNNs. We would like to have more robust experiments, but this result alone is promising.

4.1.5 Summary

We see in general that signal processing transformations lead to significant improvement in accuracy for Simple CNN, minor to significant decreases in accuracy for ResNet, and slower computation time in training. In addition, FT data greatly improves the robustness of Simple CNN to hyperparameter selection. We see that Simple CNN with FT data performs comparably to ResNet with raw data. These effects seem to correspond to how well the expressiveness of the model compares to the complexity of the transformation, with the more local wavelet transform having less of an effect than the very global fourier transform.

4.1.6 Discussion

We see that even a naïve, model agnostic inclusion of signal processing transformations can significantly improve the performance of convolutional neural networks. As efficient computation becomes more and more valuable, including signal processing methods with shallower neural networks can produce models which have many fewer parameters but perform comparably to more complex models. More

complicated architectures which intelligently integrate frequency domain data have the potential to both maintain the improvements we see on simple models while not leading to the performance drag we see on deeper models. Xu [6] explores this direction with the discrete cosine transformation and channel selection from the transformed data.

We would like to further investigate more models and models of intermediate depth to investigate the nature of the relationship between architecture depth and effectiveness of signal processing transforms. Though we are confident the effects we describe are real, they may interact differently with more or less expressive networks. Theoretically, a neural network of sufficient expressiveness should be able to learn the integral transforms we use. We would like to compare the model depth required to learn these transforms well with the model depth at which they do not positively contribute to model performance.

Lastly, we would like to have further and more robust studies of how augmenting data with signal processing transformations could help training. We suspect that it could lead to higher performances than raw data, since it includes it and also surfaces semantic information from frequencies.

Overall, we see that signal processing transforms can still present some utility in image recognition, and further research into integrating them into deep convolutional models could be fruitful.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, volume 3, page 3, 2013.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] J. Irvin, H. Sheng, N. Ramachandran, S. Johnson-Yu, S. Zhou, K. Story, R. Rustowicz, C. Elsworth, K. Austin, and A. Y. Ng. Forestnet: Classifying drivers of deforestation in indonesia using deep learning on satellite imagery. *arXiv preprint arXiv:2011.05479*, 2020.

[4] S. Mehrabkhani. Fourier transform approach to machine learning i: Fourier regression. *arXiv preprint arXiv:1904.00368*, 2019.

[5] Q. Qin, J. Li, L. Zhang, Y. Yue, and C. Liu. Combining low-dimensional wavelet features and support vector machine for arrhythmia beat classification. *Scientific reports*, 7(1):1–12, 2017.

[6] K. Xu, M. Qin, F. Sun, Y. Wang, Y.-K. Chen, and F. Ren. Learning in the frequency domain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1740–1749, 2020.