Investigation of Methods for Decoding Brain Activity

Andrej Kotevski, Jan Sobotka, Mikuláš Vanoušek
School of Computer and Communication Sciences, EPFL, Switzerland

***Abstract*—Decoding visual stimuli from brain activity is essential for understanding how sensory information is represented in the brain. Our study evaluates the performance of decoding models in relation to the availability of training data and recorded neurons, revealing critical trade-offs. Ablation experiments further highlight the impact of architectural, data, and training choices on decoding accuracy.**

I. Introduction

Recent progress in machine learning (ML), together with advances in tools for collecting large volumes of brain activity data, has enabled powerful data-driven approaches to model the brain. The dominant approach is to use ML models to characterize the stimulus-response function, i.e. predicting (*encoding*) brain activity in response to external variables, such as visual stimuli [1], [2], [3]. Yet, the inverse problem–decoding visual stimuli from brain activity–garnered less attention but is crucial for revealing which aspects of the visual scene are encoded in the brain, and is an important component for a range of applications, including visual neuro-prosthetics.

This report focuses on advancing the state-of-the-art of decoding visual scenes from neural activity recorded in early visual system. The main contributions are as follows:

1) We apply methods from [4] to a new dataset and question and motivate some of the originally proposed algorithmic decisions through ablations and more thorough hyperparameter analyses.
2) We attempt to bring pre-trained generative and classification ML models to aid in this data-scarce domain.

II. Data

The data comes from a primary visual cortex of 22 mice and was originally introduced by [5]. The primary visual cortex, also called V1, is one of the first stages of visual processing in the brain. Neurons in V1 detect low-level local visual patterns. The most salient aspect of the neural code's topological organization is the retinotopy: adjacent points of the visual field are coded by adjacent regions of V1.

Each of the 22 mouse datasets is split into training, validation, and test sets of 4500, 500, and 40 samples, respectively. A single datapoint consists of a grayscale image stimulus sampled from the ImageNet dataset [6], and associated evoked neuronal responses of around 8500 neurons from a single mouse. The number of recorded neurons differs between the individual mouse datasets. We refer the reader to [5] for additional details. In the following sections, we refer to these 22 datasets as B-1 to B-22, or collectively as B-All when merged into a single dataset.

Unless stated otherwise, all our methods, analyses, and results utilize only the mouse dataset B-6. We do so because the test samples from B-6 most closely resemble the ones shown in the original study ([5]). This choice allows for direct visual comparison and validation of our implemented baseline from the aforementioned paper, and also enables us to perform more training runs, ultimately leading to more in-depth analysis.

III. Data preprocessing

Below we describe our data preprocessing pipeline, which we apply to each of the 22 datasets. We mostly follow previous work in the area of brain activity modelling.

Images: We perform z-score normalization of the image stimuli (zero mean, unit variance). For the technique described in IV-D, we scale the images to 64×64 and 224×224 px, the Stable Diffusion Variational Autoencoder and ResNet50 input sizes, respectively. For all other techniques, we scale down the images from the original 144×256 px to 36×64 px, due to our limited computational resources.

Neuronal responses: The firing rates of different neurons vary substantially, which can be detrimental to training. To address this, we estimate the standard deviation of individual neuron activity on the training set and then rescale the neuronal responses by the inverse of these standard deviation estimates.

IV. Methods

We implemented two baseline methods, namely, a state-of-the-art inverted encoder from [5], which was originally applied to this particular mouse dataset, and a convolutional neural network. However, our main focus is on a generative adversarial network with a recently introduced input module and two variants of autoencoder-based decoders.

A. Inverted Encoder

Encoder, in this context, refers to a convolutional neural network that predicts the neuronal responses in response to images. [5] was among the first studies to also use such an encoder for the opposite task of decoding visual stimuli.

For this encoding model, the original authors used a 3-layer CNN with a linear projection readout trained on B-All. However, we found that a 4-layer architecture with a

Gaussian readout layer introduced by [2] performed slightly better, and we used this version instead[1].

Given such a pre-trained and frozen encoder model, the main idea behind the inverted encoder (InvEnc) method is to initialize an image with zero-valued or random normally distributed pixels and iteratively optimize their values through gradient descent to minimize the mean squared error between the predicted and ground-truth neural responses. Additionally, [5] applied a Gaussian blur to the image gradient at each optimization iteration to avoid high-frequency noise in the final reconstructions.

As for the other methods described below, we find optimal hyperparameters using FID on the validation set. The hyperparameter search space can be found in the Appendix.

B. Convolutional Neural Network

Our second baseline is a convolutional neural network (CNN) decoder with a fully connected input module, so-called *readin*. First, it transforms the neuronal responses using a linear layer, batch normalization, Leaky ReLU activation function, and dropout (in this order). Then, the resulting vector is unflattened into a 3D tensor of size [number of channels $\times$ height $\times$ width], which serves as input to the convolutional core of the network. This core is composed of five layers of transposed convolution, batch normalization, ReLU activation function, and dropout. After these five blocks, a convolutional layer with a single filter is applied to transform the feature map into the final reconstructed image with one channel.

It is trained with the mean squared error (MSE) training objective using the AdamW optimizer [7] with learning rate 3^{-4} and weight decay 3^{-2} for 200 epochs. Similarly to GAN in the next section, the final CNN decoder is taken to be the checkpoint that achieved the lowest FID score on the validation set. The validation set was also used for a hyperparameter search (see Appendix II for a complete list).

C. Generative Adversarial Network

The primary focus of this project is directed on the generative adversarial network (GAN) with a so-called *MEI readin* [4]. Our choice is motivated by the fact that this method showed significant improvements over the previous method [5], but its more thorough analysis has been lacking.

It adds the adversarial objective from the GAN literature as an auxiliary task for a CNN-based decoding network that is trained with the logarithmic SSIM loss [4] (other training settings and the model selection is the same as IV-B). Another major difference from vanilla GANs that synthesize images from random noise is the fact that this GAN decoder generates images conditioned only on the neuronal responses (and neuronal coordinates if available).

[1]We adopt the open-source implementation introduced as part of the SENSORIUM 2022 competition: https://github.com/sinzlab/sensorium

In addition to the reformulation of GANs and the baseline CNN IV-B, it also introduced a novel input module called *most exciting input* (MEI) readin. MEIs are stimuli, images in our case, that maximize the response of a particular neuron and have been widely used and studied in computational neuroscience. This readin module concatenates MEIs of all neurons into a 3D tensor and then pointwise multiplies it with another contextualization tensor. This contextualization is the output of a *grid network*, which takes in individual neuronal responses (and coordinates if available) and produces N independent H $\times$ W feature maps, where N is the number of neurons and H and W are the height and width of the MEIs (36×64 in our case).

We start with the same hyperparameters as in the original work [4], but explore how individual hyperparameter settings and architectural decisions impact performance. Full details of this method, including the MEI generation procedure, can be found in the original manuscript [4].

D. Transfer Learning from Pretrained Autoencoders

To alleviate the scarcity of biological data, we experiment with two pre-trained autoencoding networks. The premise is that autoencoding networks require only images for training, and we only need the limited biological data to learn the mapping from the neuronal recordings to the latent space of the autoencoder, and then use the decoder to reconstruct the image. This approach has already been explored in works such as [8] and [9].

All our hyperparameters are chosen based on the performance on the validation data of the respective datasets. See the Appendix for a complete list (III and IV).

SD-VAE: The first is the variational autoencoder of stable diffusion (SD-VAE) [10]. Inspired by [9], we map neuronal responses to SD-VAE's latent space to then use its decoder to reconstruct images. Unlike [9], which maps to both latent space and CLIP embeddings and utilizes the full diffusion pipeline, our approach focuses solely on latent space, constrained by the limited semantic information available from the mouse's primary visual cortex. The SD-VAE being pre-trained on large datasets provides a robust backbone for this task. For images of dimension 64×64, SD-VAE has a latent space of $4 \times 8 \times 8$ into which we map the neuronal responses.

We experimented with four models for this task: a ridge regression, an MLP, and two variants of CNN. Out of all of them, the MLP exhibited the best results on both the validation and test data. The MLP consists of four fully connected layers with output sizes of 4096, 2048, 1024 and 256, respectively.

ResNet50: Our second choice of a pre-trained model is motivated by the insight that brains recognize features similar to artificial neural networks (ANNs). Specifically, we selected ResNet50 as the encoder since the alignment of its layer `layer3.0.downsample.0` with neural activity

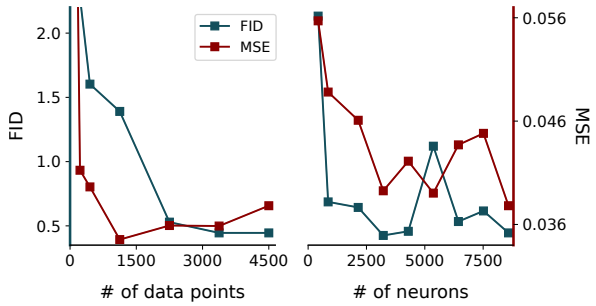

Figure 1: Relationship between performance and the amount of training data (left) and available neurons (right).

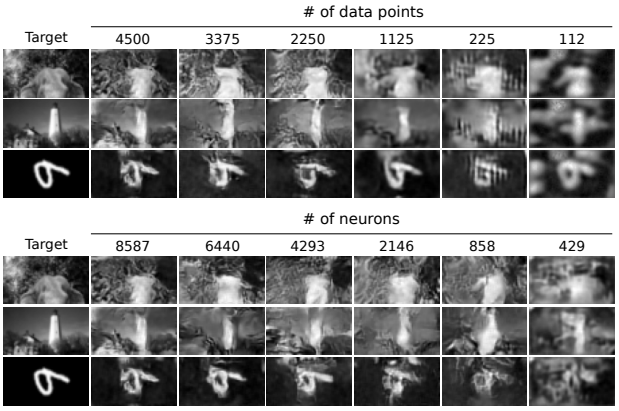

Figure 2: Reconstructed images as a function of the number of data points (top) and neurons (bottom).

in V1 was found to be the highest, as evaluated by the competition hosted on Brain-Score.org.

For the decoder, we employ nine transposed convolutions that upsample ResNet's activations from $1024 \times 14 \times 14$ to full-sized grayscale images $1 \times 224 \times 224$. Batch normalization and leaky rectified unit with negative slope 0.1 are applied after each layer except the last one to stabilize training. The training is performed on 500 randomly sampled batches from ImageNet, each containing 64 images. We use the binary cross-entropy (BCE) loss applied to the logistic function of the model's output and the min-max normalized image pixels. We found a learning rate of 0.00005 and a weight decay of 0.005 to give the best results.

We use a linear layer to map the neural recordings to $128 \times 14 \times 14$, which we then upsample to the desired number of channels using four convolutions, each preceded by a batch normalization and leaky rectified unit with negative slope 0.1. We train this readin module using Adam Optimizer with a learning rate 0.0128 and cosine scheduler, with batches of size 64 for 80 epochs. As with the VAE, we do not perform any additional fine-tuning of the autoencoder's decoder after we chain it with the readin.

V. Results and Discussion

For quantitative evaluation, we adopt the mean squared error, structural similarity index measure loss (SSIML) [4], and the Frechet Inception Distance (FID) [11].

A. Data/Neuron-Performance Relationship

First, using the same hyperparameters as in [4], we analyze the relationship between the decoding performance and 1) the amount of available training data, and 2) the number of recorded neurons. We simulate such scenarios by training the GAN decoder from scratch with varying amounts of training data and randomly subsampled neurons.

From Figure 1, we can see that the amount of data and neurons has a large impact on the performance, especially in the extremely limited data (neuron) regime. However, after the first 1000 training data points (neurons), the performance improvement starts to plateau. This shows that the GAN decoder works well in a low-data regime but may face difficulties with scaling and further improvement. It also supports the initial motivation for introducing the auxiliary adversarial objective: When combined with the main supervised decoding objective, it effectively reduces the hypothesis space of the model, directing it toward more natural-looking images while preventing collapse by the direct ground-truth decoding signal.

Next, to better understand the influence of various architectural choices of the GAN decoder, we performed a series of ablations and hyperparameter analyses.

B. Number of Channels

The biggest effect that we observed came from the number of output channels of the readin. This hyperparameter controls the width of the information channel between the neuronal recordings and the shared image generation core.

We can see in Figure 3 that the optimal number of channels is around 256, which is also close to what was used in [4] (480). The reconstructions also suggest that a smaller number of channels leads to less contrastive images, while a higher number of channels above 624 leads to hallucinated noise. This may be due to the networks memorizing the training images, hinting at a potential problem of overfitting.

C. Readin Configuration

The datasets used in [4] included neuronal coordinates along the cortical surface, which also served as inputs to the MEI readin. In our current setup, however, there are no ground truth coordinates available. We try to circumvent this issue by using the learned neuronal positions along the feature representations of the encoder core. Our motivation stems from the well-known topological organization of early visual cortices (retinotopy), which we expect to some extent to hold in the feature representation of the encoder, given that it uses the features at these coordinates for predicting the neuronal responses.

The originally proposed GAN decoder also introduced two relatively nonstandard techniques. First, it was trained

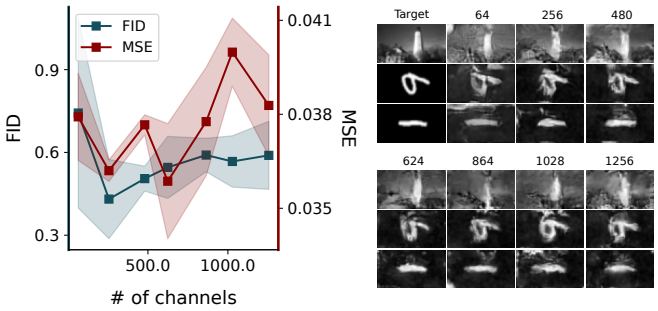

Figure 3: Impact of the number of channels on the test set performance. The plot on the left shows the average and standard deviation across three runs.

with logarithmic SSIM loss, unlike classical image generation techniques that employ MSE. Second, it transformed the neuronal responses with logarithm base 10, stating that it accelerated the training. We ablate these choices and the neuronal coordinates and show the results in Figure 4.

The plot on the left shows that both the (learned) neuronal coordinates (NC) and the response transformation (T) degrade the quality of reconstructions. This ablation shows that one could simplify the method while maintaining or improving the performance. However, this is not the case with the SSIM-based training objective: It achieves substantially lower FID and comparable MSE compared to MAE or MSE training.

Lastly, we present the final test performance of each method in Table I along with sample reconstructions in Figure 5. The dataset names in the brackets correspond to the training data. For CNN and autoencoder-based decoders, we report results only for B-6 training, while for GAN, which is the main subject of our study, we also report B-All training.

First, we can see that for GAN, there is a negative transfer from different mouse datasets (B-All). This shows that the

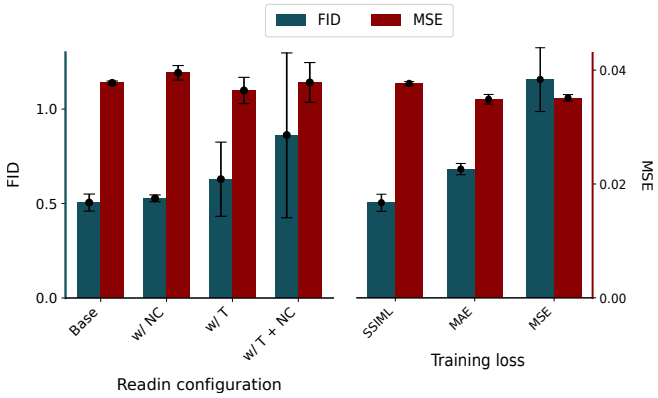

Figure 4: Test set performance with various readin configurations (left) and training objective functions (right). The average and standard deviation across three runs is shown.

Method	SSIML	MSE	FID
InvEnc	.366	.058	3.295
CNN (B-6)	.392	.066	.677
GAN (B-6)	**.323**	**.038**	**.505**
GAN (B-All)	.328	.039	.519
ResNet50 Decoder (B-6)	.468	.169	7.07
SD-VAE Decoder (B-6)	.377	.060	6.05

Table I: Quantitative results on the test set of B-6. Bold values signify the lowest (best) results.

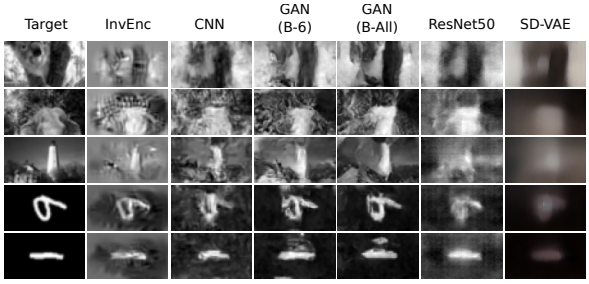

Figure 5: Reconstructed images from the test set of B-6.

individual mouse stimulus-response patterns might be too dissimilar from each other, and the GAN decoder does not have enough capacity to model all at once.

Second, in addition to the fact that GAN achieves the best quantitative results, it also matches the contrast the best (Fig. 5). This could be attributed to the SSIM-based training objective that includes a contrast term in its formulation [12].

Lastly, neither of the autoencoder-based decoders reached our best method. However, from visual inspection of the reconstructed images, we can see that the brain-aligned ResNet layer and ImageNet-only autoencoder training lead to reconstructions of much higher fidelity than those obtained using SD-VAE. It shows the importance of the training distribution and architecture of the autoencoder. We believe that better decoding results could be obtained by 1) fine-tuning the decoder end-to-end on the brain data (i.e., after we chain the readin with the decoder), and 2) utilizing the pre-trained image-to-neural-response encoder (IV-A) to add auxiliary cycle consistency objective [13] or use it for iterative response-image-response decoding (applicable also to the GAN decoder). We leave this for future work due to the time constraints of this project.

In summary, by ablating and analyzing the GAN decoder, we have shown the importance of the SSIM-based training objective and demonstrated that certain architectural choices, such as input transformations, may have been misguided. We have also shown that work needs to be done to improve the transfer learning ability of this method. Furthermore, we experimented with two autoencoder-based decoders, revealing the importance of their training distribution and proposing ways to improve it.

VI. Ethical Risks

The images paired with our brain data are sampled from ImageNet. Furthermore, one of our decoding techniques directly leverages pre-trained models on ImageNet and is further trained on this dataset. This raises the ethical concern of limited image data distribution since ImageNet has well-documented biases, which we found out through our literature research. For example, the vast majority of images originate from Europe and the United States. Furthermore, they mostly contain images from a specific socio-economic group [14]. Consequently, as with any model trained on this dataset, our techniques may perform significantly worse on images from underrepresented regions.

It would be unacceptable for such a medical device to only work in certain contexts (e.g., to be able to decode only people of a certain skin colour). For this reason, a dataset that emphasizes a wider distribution of images (geographically, socially, and ethnically) should be used. Furthermore, a rigorous evaluation of the system's performance should be conducted in individual contexts in order to verify generalization across cultural, ethnic and economic contexts. We did not perform such evaluation, due to the unavailability of brain data paired with such images.

We believe the aforementioned ethical flaw is acceptable in the context of theoretical research. However, in the event of future applications of these techniques, especially in visual neuro-prosthetics, these ethical issues become essential.

References

[1] J. Antolík, S. B. Hofer, J. A. Bednar, and T. D. Mrsic-Flogel, "Model constrained by visual hierarchy improves prediction of neural responses to natural scenes," *PLOS Computational Biology*, vol. 12, no. 6, pp. 1–22, 06 2016. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1004927

[2] K.-K. Lurz, M. Bashiri, K. Willeke, A. K. Jagadish, E. Wang, E. Y. Walker, S. A. Cadena, T. Muhammad, E. Cobos, A. S. Tolias, A. S. Ecker, and F. H. Sinz, "Generalization in data-driven models of primary visual cortex," *bioRxiv*, 2021. [Online]. Available: https://www.biorxiv.org/content/early/2021/04/06/2020.10.05.326256

[3] B. M. Li, I. M. Cornacchia, N. Rochefort, and A. Onken, "V1t: large-scale mouse v1 response prediction using a vision transformer," *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=qHZs2p4ZD4

[4] J. Sobotka, *Decoding visual stimuli from cortical activity using neural networks*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2024. [Online]. Available: https://dspace.cvut.cz/bitstream/handle/10467/115880/F8-BP-2024-Sobotka-Jan-thesis.pdf?sequence=-1&isAllowed=y

[5] E. Cobos, T. Muhammad, P. G. Fahey, Z. Ding, Z. Ding, J. Reimer, F. H. Sinz, and A. S. Tolias, "It takes neurons to understand neurons: Digital twins of visual cortex synthesize neural metamers," *bioRxiv*, 2022. [Online]. Available: https://www.biorxiv.org/content/early/2022/12/12/2022.12.09.519708

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 2009, pp. 248–255.

[7] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[8] R. Hayashi and H. Kawata, "Image reconstruction from neural activity recorded from monkey inferior temporal cortex using generative adversarial networks," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 105–109, ISSN: 2577-1655. [Online]. Available: https://ieeexplore.ieee.org/document/8616023

[9] Y. Takagi and S. Nishimoto, "High-resolution image reconstruction with latent diffusion models from human brain activity," *bioRxiv*, 2023. [Online]. Available: https://www.biorxiv.org/content/early/2023/03/11/2022.11.18.517004

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022. [Online]. Available: https://arxiv.org/abs/2112.10752

[11] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

[12] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[14] T. DeVries, I. Misra, C. Wang, and L. v. d. Maaten, "Does object recognition work for everyone?" [Online]. Available: http://arxiv.org/abs/1906.02659

VII. Appendix

A. Hyperparameter Search Space of Inverted Encoder

The reconstructed image is initialized with normally distributed pixel values, and the hyperparameter search space of the encoder inversion is as follows:

- Number of gradient steps: (300, 1000, 2000)
- Learning rate: (100, 1000, 2000)
- Standard deviation for the Gaussian blur of the image gradient: (1, 1.5, 2, 2.5)

The search space was selected based on the ranges of values the original authors used.

B. Convolutional Neural Network Architecture Details

Fully connected readin	
Latent dimension	432
Unflattened size	[3, 9, 16]
Dropout prob.	0.15
Core	
Channels	[480, 256, 256, 128, 64]
Kernel sizes	[7, 5, 5, 4, 3]
Strides	[2, 1, 1, 1, 1]
Paddings	[2, 2, 2, 1, 1]
Dropout prob.	0.35

Table II: Hyperparameters used for the CNN decoder.

C. Transfer Learning Architecture Details

Stage	Input Size	Output Size	Kernel Size	Stride	Padding
Transposed Convolutions					
1	$1024 \times 14 \times 14$	$768 \times 28 \times 28$	4	2	1
2	$768 \times 28 \times 28$	$512 \times 56 \times 56$	4	2	1
3	$512 \times 56 \times 56$	$384 \times 112 \times 112$	4	2	1
4	$384 \times 112 \times 112$	$256 \times 224 \times 224$	4	2	1
5	$256 \times 224 \times 224$	$128 \times 112 \times 112$	5	1	2
6	$128 \times 112 \times 112$	$64 \times 112 \times 112$	3	1	1
7	$64 \times 112 \times 112$	$32 \times 112 \times 112$	3	1	1
8	$32 \times 112 \times 112$	$16 \times 112 \times 112$	1	1	0
9	$16 \times 112 \times 112$	$1 \times 112 \times 112$	1	1	0

Table III: Detailed configuration of the transposed convolutions used in the architecture of the decoding network.

Stage	Input Size	Output Size	Layer Type	Kernel Size	Stride	Padding
Fully Connected and Reshape						
1	$n\_features$	$128 \times 14 \times 14$	Linear + Reshape	-	-	-
Convolutions						
2	$128 \times 14 \times 14$	$256 \times 14 \times 14$	Conv2D + BatchNorm + ReLU	5	1	2
3	$256 \times 14 \times 14$	$512 \times 14 \times 14$	Conv2D + BatchNorm + ReLU	5	1	2
4	$512 \times 14 \times 14$	$512 \times 14 \times 14$	Conv2D + BatchNorm + ReLU	1	1	0
5	$512 \times 14 \times 14$	$1024 \times 14 \times 14$	Conv2D + BatchNorm	1	1	0

Table IV: Detailed configuration of the linear and convolutional layers used in the readin module architecture.