Supplementary Material for: Visual Feature Attribution using Wasserstein GANs

Christian F. Baumgartner[1] Lisa M. Koch[2] Kerem Can Tezcan[1] Jia Xi Ang[1]
Ender Konukoglu[1] for the Alzheimer's Disease Neuroimaging Initiative

[1]Computer Vision Lab, ETH Zurich [2]Computer Vision and Geometry Group, ETH Zurich

A. Network architectures

In this section we describe the exact network architectures used for the 3D VA-GAN. We present the critic and map generator functions as Python-inspired pseudo code, which we found easier to interpret than a graphical representation. The layer parameters are specified as arguments to the layer functions. Unless otherwise specified all convolutional layers used a stride of 1x1x1 and a rectified linear unit (ReLU) non-linearity.

The architecture of the critic function $D(x)$ is shown in Fig. 1. The `conv3D_layer` function performs a regular 3D convolution without batch normalisation and the `global_averagepool3D` function performs an averaging over the spatial dimensions of the feature maps.

The architecture for the map generator function $M(x)$ is shown in Fig. 2. Here, the `conv3D_layer_bn` is a 3D convolutional layer with batch normalisation before the nonlinearity. The `deconv3D_layer_bn` learns an upsampling operation as in the original U-Net and also uses batch normalisation. Lastly, the `crop_and_concat_layer` implements the skip connections across the bottleneck by stacking the feature maps along the dimension of the channels.

Note that the architectures for the 2D experiments on synthetic data were identical, except all 3D operations were replaced by their 2D equivalents.

B. Close-up analysis of VA-GAN

In Fig. 3 we present a larger view of all three orthogonal planes for an additional subject. In order to allow for an enlarged view, we only include the results obtained by VA-GAN and the actual observed changes from MCI to AD. As before it can be seen that VA-GAN produced visual attribution maps that very closely approximate the observed deformations. In particular, we note that for this subject VA-GAN correctly predicted a smaller disease effect in the left hippocampus compared to the right hippocampus.

C. Details of MR brain data cohort

The MR brain image data used in preparation of this article were obtained from the Alzheimers Disease Neuroimaging Initiative (ADNI) database (adni.loni.usc.edu). As such, the investigators within the ADNI contributed to the design and implementation of ADNI and/or provided data but did not participate in analysis or writing of this report. A complete listing of ADNI investigators can be found at: `http://adni.loni.usc.edu/wp-content/uploads/how_to_apply/ADNI_Acknowledgement_List.pdf`.

Specifically, we used T1-weighted MR data from the ADNI1, ADNIGO and ADNI2 cohorts which were acquired in with a mixture of 1.5T and 3T scanners. The data consisted of 5770 images, acquired from 1291 subjects. The images for each subject were acquired at separate visits that were spaced in regular intervals from 6 months to one year and usually spanned multiple years. On average each subject was scanned 4.5 times. The cohort consisted of 496 female and 795 male subjects. 2839 of the images were acquired using a 1.5T magnet, the remainder using a 3T magnet. The distribution of the ages at which the images were acquired is shown in Fig. 4. We only considered images with a diagnosis of mild cognitive impairment (MCI) or Alzheimer's disease (AD).

After preprocessing we randomly divided the data into a training, testing and validation set. We performed the split on a subject basis rather than an image basis. The exact split is shown in Table 1. The table furthermore shows the distribution over the diagnoses on a image level, and the number of subjects which have undergone a conversion from MCI to AD in the examined time intervals.

The training data was used for learning the mask generator and critic parameters which minimise the cost function in Eq. 4 of the main article. The validation set was used for monitoring of the training based on the Wasserstein distance and visual examination of generated masks, and for hyperparameter tuning. The test set was used for the final qualitative and quantitative evaluation.

```
def critic(x):

    # inputs
    #    x:       an image from category c=0, or an image from category c=1
    #             plus the additive mask M(x)
    # returns
    #    logits: the critic output for x

    conv1_1 = conv3D_layer(x, num_filters=16, kernel_size=(3,3,3))

    pool1 = maxpool3D_layer(conv1_1)

    conv2_1 = conv3D_layer(pool1, num_filters=32, kernel_size=(3,3,3))

    pool2 = maxpool3D_layer(conv2_1)

    conv3_1 = conv3D_layer(pool2, num_filters=64, kernel_size=(3,3,3))
    conv3_2 = conv3D_layer(conv3_1, num_filters=64, kernel_size=(3,3,3))

    pool3 = maxpool3D_layer(conv3_2)

    conv4_1 = conv3D_layer(pool3, num_filters=128, kernel_size=(3,3,3))
    conv4_2 = conv3D_layer(conv4_1, num_filters=128, kernel_size=(3,3,3))

    pool4 = maxpool3D_layer(conv4_2)

    conv5_1 = conv3D_layer(pool4, num_filters=256, kernel_size=(3,3,3))
    conv5_2 = conv3D_layer(conv5_1, num_filters=256, kernel_size=(3,3,3))
    conv5_3 = conv3D_layer(conv5_2, num_filters=256, kernel_size=(3,3,3))

    conv5_4 = conv3D_layer(conv5_3,
                           num_filters=1,
                           kernel_size=(1,1,1),
                           nonlinearity=identity)

    logits = global_averagepool3D(conv5_4)

    return logits
```

Figure 1. VA-GAN critic architecture.

Table 1. Detailed information on data split into training, testing and validation data.

	Train	Test	Validation	Total
Num. Imag.				
MCI	2520	755	639	3914
AD	1199	399	266	1864
Total	3719	1154	905	5778
Num. Subj.				
Converters	172	51	49	272
Non-converters	653	208	158	1019
Total	825	259	207	1291

In case of interest, a list of the exact ADNI subject ID's used in the study can be found in our public code repository (`https://github.com/baumgach/vagan-code`) in the folder `data/subject_rids.txt`.

D. Alternative classifier architecture

It was suggested during the reviews that our classifier architecture with two dense layers before the final output is responsible for the poor performance of the backpropagation based saliency map techniques. It was recommended that we investigate the popular class of architectures where the final convolutions are aggregated using a global average pooling step over the spatial dimensions of the activation maps, followed by a single dense layer. Examples of this type of architecture include the works of He at al. [1] and Lin et al. [2]. In our experiments, the class activation mappings (CAM) method [7] was also using this general architecture. In theory this may abstract the data less before the final output and perhaps produce maps that can more easily identify multiple regions in the image.

To investigate this theory we repeated the synthetic experiment (outlined in Section 4.2 of the main article), but replaced the final two dense layers in our synthetic experi-

```
def map_generator(x):

    # inputs
    #    x: an image from category c=1
    # returns
    #    M: additive map M(x) such that y = x + M(x) appears to be from c=0

    # Encoder:

    conv1_1 = conv3D_layer_bn(x, num_filters=16, kernel_size=(3,3,3))
    conv1_2 = conv3D_layer_bn(conv1_1, num_filters=16, kernel_size=(3,3,3))

    pool1 = maxpool3D_layer(conv1_2)

    conv2_1 = conv3D_layer_bn(pool1, num_filters=32, kernel_size=(3,3,3))
    conv2_2 = conv3D_layer_bn(conv2_1, num_filters=32, kernel_size=(3,3,3))

    pool2 = maxpool3D_layer(conv2_2)

    conv3_1 = conv3D_layer_bn(pool2, num_filters=64, kernel_size=(3,3,3))
    conv3_2 = conv3D_layer_bn(conv3_1 num_filters=64, kernel_size=(3,3,3))

    pool3 = maxpool3D_layer(conv3_2)

    # Bottleneck:

    conv4_1 = conv3D_layer_bn(pool3, num_filters=n128, kernel_size=(3,3,3))
    conv4_2 = conv3D_layer_bn(conv4_1, num_filters=128, kernel_size=(3,3,3))

    # Decoder:

    upconv3 = deconv3D_layer_bn(conv4_2, kernel_size=(4,4,4), strides=(2,2,2), num_filters=64)
    concat3 = crop_and_concat_layer([upconv3, conv3_2])

    conv5_1 = conv3D_layer_bn(concat3, num_filters=64, kernel_size=(3,3,3))
    conv5_2 = conv3D_layer_bn(conv5_1, num_filters=64, kernel_size=(3,3,3))

    upconv2 = deconv3D_layer_bn(conv5_2, kernel_size=(4,4,4), strides=(2,2,2), num_filters=32)
    concat2 = crop_and_concat_layer([upconv2, conv2_2])

    conv6_1 = conv3D_layer_bn(concat2, num_filters=32, kernel_size=(3,3,3))
    conv6_2 = conv3D_layer_bn(conv6_1, num_filters=32, kernel_size=(3,3,3))

    upconv1 = deconv3D_layer_bn(conv6_2, kernel_size=(4,4,4), strides=(2,2,2), num_filters=16)
    concat1 = crop_and_concat_layer([upconv1, conv1_2])

    conv8_1 = conv3D_layer_bn(concat1, num_filters=16, kernel_size=(3,3,3))

   M = conv3D_layer(conv8_1,
                    num_filters=1,
                    kernel_size=(3,3,3),
                    nonlinearity=identity)

    return M
```

Figure 2. VA-GAN map generator architecture.

ments by a global average pooling and a single dense layer. After full convergence of the network from the main article and the alternative architecture, we obtained the saliency maps shown in Fig. 5. In addition to the integrated gradients method [6] already shown in the main article, here we also show the results for normal backprop [4] and guided backprop [5]. It can be observed that indeed, with the alternative architecture, normal and guided backprop manage to correctly attribute some of the pixels of the peripheral box, albeit very faintly (emphasised with white arrows in Fig 5). However, regardless of the architecture the classifier appears to focus only on the pixels of one of the edges,

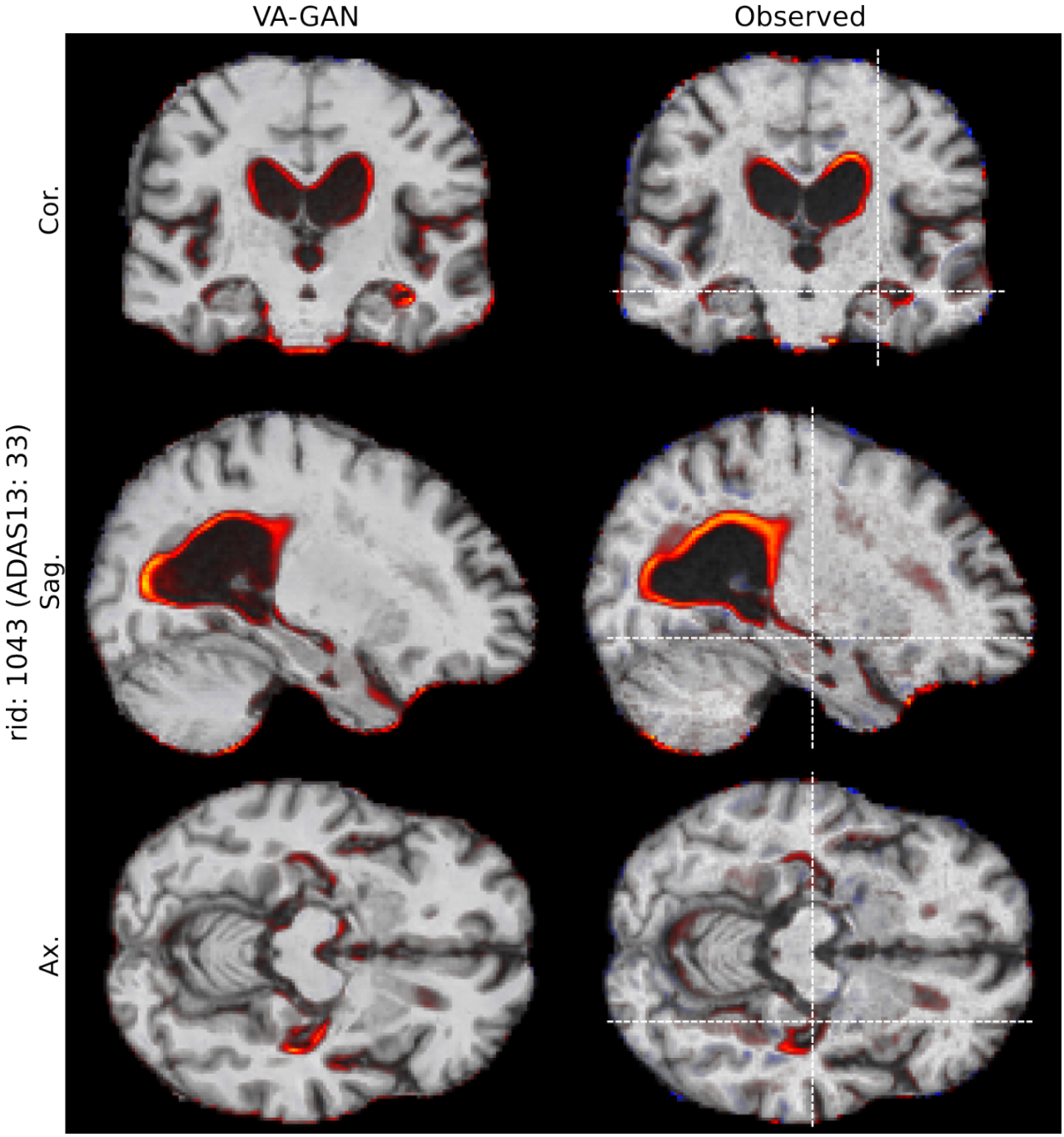

Figure 3. Coronal, sagittal and axial views of the predicted and observed disease effect maps for an additional subject. The location of the planes is indicated by dotted white lines in the right column. In order to allow for an enlarged view, only the predictions obtained by VA-GAN are shown. The ADNI rid and the ADAS13 score for this subject are reported on the left-hand side.

which is only subset of the features characterising this class. Note that the orientation of the attributed edges depends on the random initialisation of the network.

Nevertheless, the feature attribution maps obtained using the backprop-based techniques are not of comparable quality to the maps produced by our proposed VA-GAN method. For emphasis we show the corresponding feature attribution map produced with VA-GAN plus two more samples in Fig. 6.

To conclude, we would like to note that from the point of view of saliency maps, (1) two dense layers or (1) average pooling followed by a dense layer, are conceptually

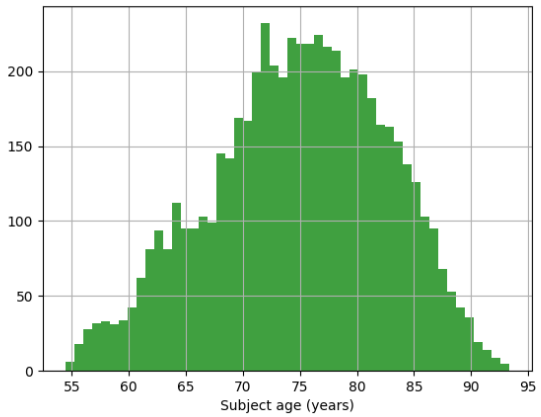

Figure 4. Histogram of the subject age of all ADNI images used in this work. The mean age was 74.89 years, with a standard deviation of 7.70.

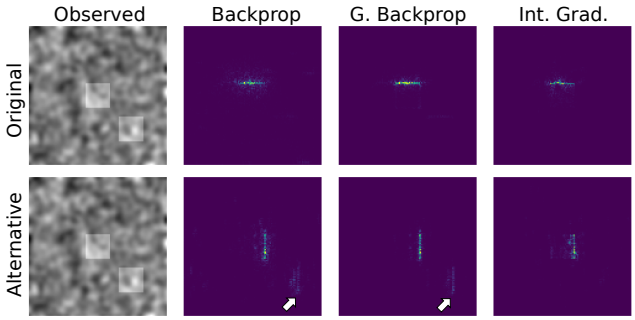

Figure 5. Saliency maps obtained using simple backpropagation, guided backpropagation and integrated gradients for two different network architectures: (1) the original architecture from the synthetic experiments (Section 4.2) in the main article, (2) an alternative architecture with a global average pooling layer followed by a single dense layer before the final classification output. The white arrows in the second row highlight very faint attributions of the second box.

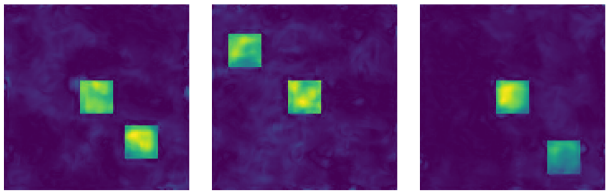

Figure 6. Visual feature attribution maps obtained using our proposed VA-GAN method. The first sample corresponds to the input image in Fig. 5. The other two images correspond to other random input images.

similar. In both cases the final prediction aggregates information from multiple receptive fields covering the whole image. Therefore, it is not surprising that the two networks behave similarly. As outlined in the work of Shwartz-Ziv et al. [3] the optimisation of neural network classifiers results in a trade off between compression of input features and predictive accuracy. In both networks, the final prediction has access to all features in the image and thus has the potential to compress away features that are redundant for classification (such as one of the two boxes).

References

[1] 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.

[2] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[3] R. Shwartz-Ziv and N. Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[4] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[5] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[6] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[7] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.