DETECTION OF PRE-MICRORNA WITH CONVOLUTIONAL NEURAL NETWORKS

Jorge Cordero, Vlado Menkovski and Jens Allmer

1 CNN architectures

The networks described in Supplementary Tables 1, 2, and 3 were implemented using the Keras deep learning framework, and their code is available at `https://github.com/jacordero/deepmir/`. The code includes scripts to design and train VGG-Style, Inception-Style, and ResNet-Style CNN models.

Supplementary Table 1: CNNs based on the VGG architecture [1].

VGG-Style-1M		**VGG-Style-2M**		**VGG-Style-3M**		**VGG-Style-4M**	
Layer	Params	Layer	Params	Layer	Params	Layer	Params
Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)
Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)	Convolutional	(3×3, 48×)
Maxpool	(2×2)	Maxpool	(2×2)	Maxpool	(2×2)	Maxpool	(2×2)
Dropout	0.25	Dropout	0.25	Dropout	0.25	Dropout	0.25
Flatten		Convolutional	(3×3, 60×)	Convolutional	(3×3, 60×)	Convolutional	(3×3, 48×)
FC	(FU×)	Convolutional	(3×3, 60×)	Convolutional	(3×3, 60×)	Convolutional	(3×3, 48×)
Dropout	0.5	Maxpool	(2×2)	Maxpool	(2×2)	Maxpool	(2×2)
FC	(2×)	Dropout	0.25	Dropout	0.25	Dropout	0.25
Softmax		Flatten		Convolutional	(3×3, 72×)	Convolutional	(3×3, 72×)
		FC	(FU×)	Convolutional	(3×3, 72×)	Convolutional	(3×3, 72×)
		Dropout	0.5	Maxpool	(2×2)	Maxpool	(2×2)
		FC	(2×)	Dropout	0.25	Dropout	0.25
		Softmax		Flatten		Convolutional	(3×3, 84x)
				FC	(FU×)	Convolutional	(3×3, 84x)
				Dropout	0.5	Maxpool	(2×2)
				FC	(2×)	Dropout	0.25
				Softmax		Flatten	
						FC	(FU×)
						Dropout	0.5
						FC	(2×)
						Softmax	

The table shows the structure of VGG-Style networks with up to four convolutional modules (VGG-Style-4M). Each convolutional module is made of two convolutional layers using filter size of 3×3 and the same number of filters (48, 60, 72, or 84) followed by a maxpool layer and a dropout layer. The end of the networks consists of a fully connected module made of a fully connected layer containing a certain number of fully connected units FU (32, 64, 128, or 256) followed by a dropout layer, another fully connected layer with two units, and a softmax layer. In total, we can create 16 different CNNs for a given dataset: four different VGG-Style networks with four different values for the FU parameter.

2 CNNs performance

Table 4 shows the accuracy of the best ten CNN models trained using transfer learning and their corresponding versions without pre-training. Even though 48 different models were trained following the VGG-Style, Inception-Style, and ResNet-Style defined in the tables above, only VGG-Style models belong to the top ten models. The models are sorted according to their average accuracy on both datasets. Tables 5 to 7 show the accuracy of VGG-Style, Inception-Style, and ResNet-Style models respectively. The accuracy of the best models is emphasized in bold.

Supplementary Table 2: CNNs based on the Inception architecture [2].

Inception-Style-1M		**Inception-Style-2M**		**Inception-Style-3M**		**Inception-Style-4M**	
Layer/Module	Params	Layer/Module	Params	Layer/Module	Params	Layer/Module	Params
Convolutional	(64×, 3×3)	Convolutional	(64×, 3×3)	Convolutional	(64×, 3×3)	Convolutional	(64×, 3×3)
Inception	(CK×)	Inception	(CK×)	Inception	(CK×)	Inception	(CK×)
Flatten		Inception	(CK×)	Inception	(CK×)	Inception	(CK×)
FC	(2×)	Flatten		Inception	(CK×)	Inception	(CK×)
Softmax		FC	(2×)	Flatten		Inception	(CK×)
		Softmax		FC	(2×)	Flatten	
				Softmax		FC	(2×)
						Softmax	

The first layer of each network is a convolutional layer with 64 filters of shape 3×3 used to extract low level features. The Inception modules, used to extract spatial patterns at different resolutions, have the structure shown in Figure 1. This Inception module has convolutional layers with different filter shape that the ones presented in the original GoogLeNet, the maxpool layer also differs. All the convolutional layers inside the Inception modules have the same number of filters (CK), which can take four different values: 8, 12, 16, and 20. In total, we can create 16 different CNNs for a given dataset: four different Inception-Style networks with four different values for the CK parameter.

Supplementary Table 3: CNNs based on the ResNet architecture [3] using fully-preactivated units [4].

ResNet-Style-1M		**ResNet-Style-2M**		**ResNet-Style-3M**		**ResNet-Style-4M**	
Layer/Module	Params	Layer/Module	Params	Layer/Module	Params	Layer/Module	Params
Convolutional	(32×, 3×3)	Convolutional	(32×, 3×3)	Convolutional	(32×, 3×3)	Convolutional	(32×, 3×3)
ResNet	(32×, 3×3)	ResNet	(32×, 3×3)	ResNet	(32×, 3×3)	ResNet	(32×, 3×3)
GPA		ResNet	(32×, 3×3)	ResNet	(32×, 3×3)	ResNet	(32×, 3×3)
FC	(2×)	GPA		ResNet	(32×, 3×3)	ResNet	(32×, 3×3)
Softmax		FC	(2×)	GPA		ResNet	(32×, 3×3)
		Softmax		FC	(2×)	GPA	
				Softmax		FC	(2×)
						Softmax	

The first layer of each network is a convolutional layer with 32 filters of shape 3×3 used to extract low level features. The ResNet modules have convolutional layers with a filter shape of 3×3. The number of filters in the convolutional layers is given by the parameter CK that can take four different values: 20, 24, 28, and 32. The GPA layer corresponds to a global average pooling layer that maps 3D feature maps to 1D vectors. The units of such vectors are then connected to the softmax units to produce the output of the network. In total, we can create 16 different CNNs for a given dataset: four different ResNet-Style networks with four different values for the CK parameter.

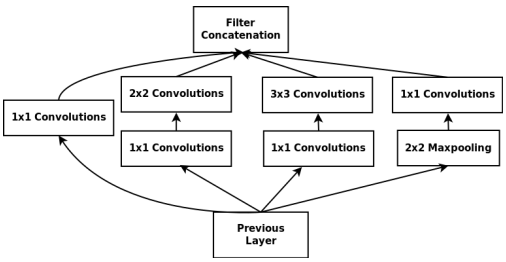

Figure 1: Inception module designed for pre-miRNA detection. This module follows a similar structure to the original Inception module presented in the GoogLeNet network, but it uses different filter shapes for the convolutional and maxpool layers. The convolutional layers have filters with shape 1×1, 2×2, and 3×3, while the maxpool layer has filter shape of 2×2.

Supplementary Table 4: Accuracy of top ten pre-trained CNN models.

CNN	With pre-training	Without pre-training
VGG-Style-3M-256FU	**0.9517**	0.9171
VGG-Style-1M-128FU	0.9472	0.8934
VGG-Style-3M-128FU	0.9472	0.8907
VGG-Style-1M-64FU	0.9454	0.8889
VGG-Style-3M-64FU	0.9454	0.8807
VGG-Style-2M-256FU	0.9426	0.9180
VGG-Style-4M-128FU	0.9426	0.9199
VGG-Style-1M-32FU	0.9417	0.9044
VGG-Style-1M-256FU	0.9408	0.8962
VGG-Style-4M-256FU	0.9408	0.9089

Among all the CNN models based on VGG-Style, Inception-Style, and ResNet-Style architectures, only VGG-Style models belong to the top ten CNNs. The corresponding non pre-trained models are also included to observe the benefits of pre-training.

Supplementary Table 5: Accuracy of 16 VGG-Style networks trained with and without pre-training.

CNN	Without pre-training				With pre-training			
	32FU	64FU	128FU	256FU	32FU	64FU	128FU	256FU
VGG-Style-1M	0.9044	0.8889	0.8934	0.8962	0.9417	0.9454	0.9472	0.9408
VGG-Style-2M	0.9162	0.9144	**0.9199**	0.9180	0.9390	0.9390	0.9381	0.9426
VGG-Style-3M	0.8989	0.8807	0.8907	0.9171	0.9362	0.9454	0.9472	**0.9517**
VGG-Style-4M	0.9162	0.9089	**0.9199**	0.9089	0.9244	0.9381	0.9426	0.9408

The *modhsa* test dataset is used to evaluate the performance of all trained networks. For pre-trained networks, VGG-Style-3M-256FU has the best performance, while for non pre-trained networks, VGG-Style-2M-128FU and VGG-Style-4M-128FU achieve the best performance.

Supplementary Table 6: Accuracy of 16 Inception-Style networks trained with and without pre-training.

CNN	Without pre-training				With pre-training			
	8CK	12CK	16CK	20CK	8CK	12CK	16CK	20CK
Inception-Style-1M	0.8843	0.8880	0.5000	0.5000	0.9311	0.9299	0.9344	0.5000
Inception-Style-2M	0.8907	0.8807	**0.8953**	0.8852	**0.9353**	0.5000	0.9290	0.9290
Inception-Style-3M	0.8834	0.8916	0.8880	0.5000	0.9235	0.9271	0.9335	0.9281
Inception-Style-4M	0.8871	0.8843	0.8934	0.8862	0.9281	0.9235	0.9262	0.9226

The *modhsa* test dataset is used to evaluate the performance of all trained networks. For pre-trained networks, Inception-Style-2M-8CK has the best performance, while for non pre-trained networks, Inception-Style-2M-16CK achieves the best performance.

Supplementary Table 7: Accuracy of 16 ResNet networks trained with and without pre-training.

CNN	Without pre-training				With pre-training			
	20CK	24CK	28CK	32CK	20CK	24CK	28CK	32CK
ResNet-Style-1M	0.5009	0.9098	0.9044	0.9135	0.8242	**0.9208**	0.5082	0.5337
ResNet-Style-2M	0.5055	0.5000	0.5000	0.5000	0.8780	0.6366	0.6967	0.8652
ResNet-Style-3M	0.5893	**0.9199**	0.9044	0.5310	0.9117	0.8470	0.6894	0.9117
ResNet-Style-4M	0.9062	0.8233	0.8925	0.9123	0.5729	0.8525	0.5337	0.7570

The *modhsa* test dataset is used to evaluate the performance of all trained networks. For pre-trained networks, ResNet-Style-1M-24CK has the best performance, while for non pre-trained networks, ResNet-Style-3M-24CK achieves the best performance.

References

[1] Simonyan, K. and Zisserman, A. (2014) Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR,* **abs/1409.1556**.

[2] Szegedy, C., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (June, 2015) Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 1–9.

[3] He, K., Zhang, X., Ren, S., and Sun, J. (2015) Deep Residual Learning for Image Recognition. *CoRR,* **abs/1512.03385**.

[4] He, K., Zhang, X., Ren, S., and Sun, J. (2016) Identity Mappings in Deep Residual Networks. *CoRR,* **abs/1603.05027**.