

layers which perform the convolution operation between the input values and convolution filters (matrix of weights) that are learned over many gradient update iterations during the training.

[00131] Let m be the filter size and W be the matrix of weights, then a convolution layer performs a convolution of the W with the input X by calculating the dot product $W \cdot x + b$, where x is an instance of X and b is the bias. The step size by which the convolution filters slide across the input is called the stride, and the filter width m is called the receptive field. A same convolution filter is applied across different positions of the input, which reduces the number of weights learned. It also allows location invariant learning, i.e., if an important pattern exists in the input, the convolution filters learn it no matter where it is in the sequence. Additional details about the convolutional neural network **612** can be found in I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “CONVOLUTIONAL NETWORKS,” Deep Learning, MIT Press, 2016; J. Wu, “INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS,” Nanjing University, 2017; and N. ten DIJKE, “Convolutional Neural Networks for Regulatory Genomics,” Master’s Thesis, Universiteit Leiden Opleiding Informatica, 17 June 2017, the complete subject matter of which is expressly incorporated herein by reference in its entirety.

[00132] After processing the input sequence **602**, the convolutional neural network **612** produces an intermediate convolved feature **622** as output. The concatenator **112** concatenates (*) the intermediate convolved feature **622** with one or more metadata features **626** discussed above. Concatenation can occur across the row dimension or the column dimension. For example, concatenating 1x465 outputs of the intermediate convolved feature **622** and 1x49 metadata features **626** would produce a feature sequence **634** of 1x514 values. The result of the concatenation is a feature sequence **634**, which is stored in the database **122**.

[00133] The feature sequence **634** is fed as input to the fully-connected neural network (FCNN) **674**. The fully-connected neural network **674** comprises fully-connected layers - each neuron receives input from all the previous layer’s neurons and sends its output to every neuron in the next layer. This contrasts with how convolutional layers work where the neurons send their output to only some of the neurons in the next layer. The neurons of the fully-connected layers are optimized over many gradient update iterations during the training. Additional details about the fully-connected neural network **674** can be found in I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “CONVOLUTIONAL NETWORKS,” Deep Learning, MIT Press, 2016; J. Wu, “INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS,” Nanjing University, 2017; and N. ten DIJKE, “Convolutional Neural Networks for Regulatory Genomics,” Master’s Thesis, Universiteit Leiden Opleiding Informatica, 17 June 2017, the complete subject matter of which is expressly incorporated herein by reference in its entirety.

[00134] A classification layer **684** of the fully-connected neural network **674** outputs classification scores **694** for likelihood that the variant is a somatic variant, a germline variant, or noise. The classification layer **684** can be a softmax layer or a sigmoid layer. The number of classes and their type can be modified, depending on the implementation. As discussed above, having the noise category improves classification along the somatic and germline categories.

[00135] In other implementations, the metadata features **626** can be fed directly to the convolutional neural network **612** and encoded into the input sequence **602** or fed separately, but simultaneously with the input sequence **602** or fed separately, but before/after the input sequence **602**.

[00136] FIG. 7 shows an algorithmic example **700** of variant classification performed by the variant classifier **104**. In the illustrated implementation, the convolution neural network (CNN) **612** has two convolution layers and the fully-connected neural network (FCNN) **674** has three fully-connected layers. In other implementations, the variant classifier **104**, and its convolution neural network **612** and fully-connected neural network **674**, can have additional, fewer, or different parameters and hyperparameters. Some examples of parameters are number of convolution layers, number of

batch normalization and ReLU layers, number of fully-connected layers, number of convolution filters in respective convolution layers, number of neurons in respective fully-connected layers, number of outputs produced by the final classification layer, and residual connectivity. Some examples of hyperparameters are window size of the convolution filters, stride length of the convolution filters, padding, and dilation. In the discussion below, the term “layer” refers to an algorithm implemented in code as a software logic or module. Some examples of layers can be found in Keras™ documentation available at <https://keras.io/layers/about-keras-layers/>, the complete subject matter of which is expressly incorporated herein by reference in its entirety.

[00137] A one-hot encoded input sequence **702** is fed to a first convolution layer **704** of the convolutional neural network (CNN) **612**. The dimensionality of the input sequence **702** is 41, 5, where 41 represents the 41 bases in the input sequence **702** with a particular variant at a center target position flanked by 20 bases on each side, and 5 represents the 5 channels A, T, C, G, N used to encode the input sequence **702** and illustrated in **FIG. 3**.

[00138] The first convolution layer **704** has 25 filters, each of which convolves over the input sequence **702** with a window size of 7 and stride length of 1. The convolution is followed by batch normalization and ReLU nonlinearity layers **712**. What results is an output (feature map) **714** of dimensionality 25, 35. Output **714** can be regarded as the first intermediate convolved feature.

[00139] Output **714** is fed as input to a second convolution layer **722** of the convolutional neural network **612**. The second convolution layer **722** has 15 filters, each of which convolves over the output **714** with a window size of 5 and stride length of 1. The convolution is followed by batch normalization and ReLU nonlinearity layers **724**. What results is an output (feature map) **732** of dimensionality 15, 31. Output **732** can be regarded as the second intermediate convolved feature and also the final output of the convolutional neural network **612**.

[00140] In order to concatenate the output **732** with the metadata features **742** and also to allow downstream processing by the fully-connected neural network (FCNN) **674**, the output **732** is flattened by a flattening layer **734**. Flattening includes vectorizing the output **732** to have either one row or one column. That is, by way of example, converting the output **732** of dimensionality 15, 31 into a flattened vector of dimensionality 1, 465 (1 row and $15 \times 31 = 465$ columns).

[00141] The metadata features **742**, correlated with the particular variant, have a dimensionality of 49, 1. A concatenation layer **744** concatenates the metadata features **742** with the flattened vector derived from the output **732**. What results is a flattened metadata feature output **752** of dimensionality 1, 49. Output **752** can be regarded as the feature sequence. Concatenation of the flattened vector **732** and the flattened output **752** produces a vector 1, 514.

[00142] The output **752** is then fed as input to the fully-connected neural network (FCNN) **674**. The fully-connected neural network **674** has three fully-connected layers **754**, **764**, and **774**, each succeeded by pairs **762**, **772**, and **782** of batch normalization and ReLU nonlinearity layers. The first fully-connected layer **754** has 512 neurons, which are fully connected to 512 neurons in the second fully-connected layer **764**. The 512 neurons in the second fully-connected layer **764** are fully connected to 256 neurons in the third fully-connected layer **774**.

[00143] The classification layer **784** (e.g., softmax) has 3 neurons which output the 3 classification scores or probabilities **792** for the particular variant being somatic, germline, or noise.

[00144] In other implementations, the metadata features **742** can be fed directly to the convolutional neural network **612** and encoded into the input sequence **702** or fed separately, but simultaneously with the input sequence **702** or fed separately, but before/after the input sequence **702**.

Transfer Learning

[00145] **FIG. 8** depicts one implementation of training the variant classifier **104** according to a transfer learning strategy **800**, followed by evaluation and testing of the trained variant classifier **104**. Transfer learning strategy **800** involves pre-training **802** the variant classifier **104** on a base dataset **812** (e.g., TCGA) and task (variant