Deep SNP: An End-to-end Deep Neural Network with Attention-based Localization for Break-point Detection in SNP Array Genomic data

Hamid Eghbal-zadeh [*1] Lukas Fischer [*2] Niko Popitsch [3] Florian Kromp [3] Sabine Taschner-Mandl [3]
Khaled Koutini [1] Teresa Gerber [3] Eva Bozsaky [3] Peter F. Ambros [3] Inge M. Ambros [3] Gerhard Widmer [1]
Bernhard A. Moser [2]

Abstract

Diagnosis and risk stratification of cancer and many other diseases require the detection of genomic breakpoints as a prerequisite of calling copy number alterations (CNA). This, however, is still challenging and requires time-consuming manual curation. As deep-learning methods outperformed classical state-of-the-art algorithms in various domains and have also been successfully applied to life science problems including medicine and biology, we here propose Deep SNP, a novel Deep Neural Network to learn from genomic data. Specifically, we used a manually curated dataset from 12 genomic single nucleotide polymorphism array (SNPa) profiles as truth-set and aimed at predicting the presence or absence of genomic breakpoints, an indicator of structural chromosomal variations, in windows of 40,000 probes. We compare our results with well-known neural network models as well as Rawcopy though this tool is designed to predict breakpoints and in addition genomic segments with high sensitivity. We show, that Deep SNP is capable of successfully predicting the presence or absence of a breakpoint in large genomic windows and outperforms state-of-the-art neural network models. Qualitative examples suggest that integration of a localization unit may enable breakpoint detection and prediction of genomic segments, even if the breakpoint coordinates were not provided for network training. These results warrant further evaluation of DeepSNP for breakpoint localization and subsequent calling of genomic segments.

1. Introduction

Copy-number alterations (CNA) such as losses, gains or amplifications of DNA sequences result in changes in the copy number status (CNS) of the respective genomic regions. If CNAs occur within chromosomes, this results in segmental CNAs and they can range in size from a few base pairs to whole chromosome arms and are, for example, used to molecularly diagnose and/or stratify cancer patients into risk-groups, a pre-requisite to allocate patients to appropriate treatment protocols. Accurate identification of CNAs is thus critical for studying the pathogenesis of cancer and many other diseases.

Today, despite emerging new technologies such as whole-genome sequencing (WGS), microarrays and in particular single nucleotide polymorphism array (SNPa) are a common choice for copy number analyses in clinical routine. Reasons for this arguably include their simplicity, robustness and good cost-benefit ratio. SNPa interrogate a patient genome at defined genomic positions using copy number and allele-specific oligonucleotide probes.

A core step of copy-number analyses is the segmentation of patient genomes into segments of equal copy number. Such genomic segments are defined by their endpoints (aka breakpoints), i.e. transitions in the copy number state (CNS) of two adjacent genomic segments, and can be determined by considering neighboring probes of a SNPa. These probes provide information about the relative abundance of DNA at a particular genomic position (i.e., its copy-number) and about the directly related ratio of two different alleles at polymorphic (SNP) positions (the B-allele frequency) (LaFramboise, 2009).

Accurate copy number calling from these data requires breakpoint detection, but is challenging for multiple reasons including mosaicism, technical noise, repetitiveness of the genome, and technology intrinsic biases such as guanine-cytosine (GC) bias (DNA fragments that are very rich/poor in G and C bases show different hybridization/amplification characteristics, cf. (Benjamini & Speed, 2012)) or probe cross-hybridization.

[*] Equal contribution [1] Institute of Computational Perception, Johannes Kepler University, Linz, Austria [2] Software Competence Center Hagenberg (SCCH), Hagenberg, Austria [3] Children's Cancer Research Institute (CCRI), Vienna, Austria. Correspondence to: Hamid Eghbal-zadeh <hamid.eghbal-zadeh@jku.at>.

Joint ICML and IJCAI 2018 Workshop on Computational Biology
Copyright 2018 by the author(s).

In this paper we propose a novel end-to-end deep neural network namely Deep SNP, specialized to process raw SNP array genomic data and to predict breakpoints therein. Deep SNP is capable of processing very long stretches of copy number data by incorporating dilated convolution layers and it benefits from state-of-the-art feature learning architectures. In addition, it can learn long genomic distance relations in its distance embedding space using recurrent layers. Finally, by integrating attention-based localization layers, Deep SNP is capable of accurately pinpointing breakpoints without the need of using accurate labels for training and reduce the recall and increase breakpoint calling precision within the given genomic windows of 40,000 probes, compared to the general purpose state-of-the-art deep models. Though direct comparison is difficult, Deep SNP also compares well to biological tools such as Rawcopy (Mayrhofer et al., 2016) in predicting the presence of breakpoints.

2. Related Work

Over time, many different algorithms for breakpoint detection have been proposed, mostly relying on statistical methods including Hidden Markov Models (HMM) (Wang et al., 2007; Marioni et al., 2006), Bayesian approaches (Pique-Regi et al., 2008; Zhang & Gerstein, 2010) or circular binary segmentation (CBS; (Olshen et al., 2004)), a simple recursive method that was shown to outperform other approaches in terms of sensitivity and false discovery rate (FDR; (Zhao et al., 2013)). Briefly, CBS starts by considering a whole chromosome and recursively segments it by applying a simple statistical test (maximal t-statistic) for change point detection to these segments. CBS stops when no more change points can be found in the current segmentation (Venkatraman & Olshen, 2007). HMM and CBS based solutions are implemented in many popular copy number variation (CNV) tools such as ChAS, Nexus or Rawcopy and provide relatively accurate and stable copy-number calls.

Current algorithms perform well in routine analysis, when DNA quality and quantity are sufficient and tumor cell content is high and the respective SNPa data shows relatively „clean" profiles. In practice, however, SNPa profiles are often noisy and reasons for this include cross-linked and/or fragmented DNA due to tissue preparation procedures such as formalin fixation or unusually long storage of tissue/DNA. Furthermore, when DNA amounts are limited, such as in cell-free DNA extractions from liquid biopsies, low DNA input and contamination with non-tumor DNA can also contribute to the noise in the data. With current algorithms, noisy and/or subclonal/contaminated samples result often in reduced segmentation accuracy, mainly due to increased false positive rates, and/or high fragmentation (i.e. many small neighboring segments of alternating copy number). Such segments require manual curation which is work intensive and error prone. Thus, there is a need to develop algorithms predicting breakpoints and genomic segments with higher precision from these data.

Deep learning methods outperformed classical state-of-the-art algorithms across various domains including image classification, speech recognition, language translation or document analysis and were recently also successfully applied to various problems in the life science domain (Angermueller et al., 2016). Deep learning architectures can learn unknown or hidden relationships from highly complex and noisy data if a sufficient amount of training data is available, even if only weak labels can be provided (Ching, 2018). DeepVariant, a deep learning architecture to predict SNVs or small insertions/deletions (indels) in WGS data was proposed by (Poplin et al., 2016). Although results show that DeepVariant can learn the statistical relationship between aligned reads and true SNVs/indels over various sequencing technologies, the architecture is not directly transferable to be applied to call CNVs from SNPa data as technology and thus data structure and resolution differ substantially. Moreover, the architecture is a pure classification approach relying on currently available frameworks calling candidate predictions. Thus, DeepVariant is not suitable to further call CNVs. Most recently, (Gupta & Rush, 2017) proposed to use dilated convolution on regulatory marker locations from ENCODE (Consortium, 2012) to model long-distance genomic dependencies. They showed that dilated convolution can outperform LSTM (Hochreiter & Schmidhuber, 1997) recurrent models. Nevertheless they did not evaluate their method on detection of rare genomic events such as breakpoints. In addition, their method requires long input sequences and does not have a localization ability.

3. Data Collection and Preprocessing

For our initial tests we analyzed 12 samples of 5 neuroblastoma patients (material was either tumor, or disseminated tumor cells (DTCs)) on the Affymetrix CytoScan HD SNP array platform (Ambros et al., 2014). The resulting CEL files contain $\sim$ 2.8 Mio raw array intensity values which were converted into normalized log ratio (LRR) and B-allele frequency values (BAF) using Rawcopy, an open R package for processing Affymetrix microarray data (Mayrhofer et al., 2016). Rawcopy first calculates raw LRR and BAF values according to the following formulas where $\overline{A}$ and $\overline{B}$ are mean intensities of the respective SNPa probes (see (Mayrhofer et al., 2016) for details):

$$LRR = log_2\sqrt{\overline{A^2} + \overline{B^2}} \tag{1}$$

$$BAF = \frac{\overline{B}}{\overline{A} + \overline{B}} \tag{2}$$

These two measures are then further normalized using observed per-probe value distributions derived from a large set of reference samples. LRR values are furthermore corrected for fragment length and GC content. Finally, Rawcopy calls breakpoints using an allele-specific CBS algorithm and outputs text files containing normalized LRR and BAF values as well as called copy number segments. These resulting data files contained $2,819,443$ LRR values and around $480,000$ BAF values each (Figure 1, (a)).

To create the ground truth, we manually curated the Rawcopy predictions using our in-house editor Varan-GIE[1], taking additional data files (such as related datasets, mappability tracks and annotations of common CNVs) into account. Highly fragmented segments often occur in noisy profiles as artifacts predicted by Rawcopy and need to be merged. In case of false negative segment predictions, additional segments have to be generated and added to the final segmentation. More frequently, due to a high sensitivity but lower specificity of Rawcopy predictions, false positive segments have to be removed. The manually curated segments were called by experienced biologists and always double-checked by at least one additional person (according to the lab routine) and considered as the truth-set in the following.

To train our network, we combined the Rawcopy output, our manual curation set and additional genome-wide data to create genome-wide data files that contain the following data: 1) Genomic position (chromosome + offset) of the respective SNPa probe, 2) Rawcopy LRR value, 3) Rawcopy BAF (-1 encoding missing values), 4) Encoded truth-set copy number state (normal, loss, gain, amplification) at the probe position, and 5) Encoded Rawcopy copy number state at the probe position (used in our evaluation).

3.1. Data Preparation for Deep Learning

In this section, we explain the details of data preparation we have done to feed the SNPa data to our deep neural networks. The samples of SNPa are very long (~ 2.8 Mio. values) and can not be directly fed to a neural network to process. Therefore, we designed a windowing scheme to create training and validation data for the neural network systems. Using this procedure, based on the truth-set of copy number state we select positive and negative windows of probes and used them as training and validation data. A window of n probes is considered a positive window, if within that window the copy number state changes at least once. Also a window of n probes is considered a negative window, if the copy number state does not change at all within that window. The change of copy number states (copy number state transition) represents a breakpoint, therefore from now on, a positive window will be referred to as a window with at least one breakpoint.

For each genomic sample, we select the positive windows as follows: First we locate the position of all breakpoints in that sample based on the truth-set. Second, a window of n probes is centered at all the known breakpoint positions one by one. Due to the possible small distance between neighboring breakpoints, positive windows can contain multiple breakpoints.

To select negative windows, we randomly select (non-overlapping) windows with n consecutive probes that do not contain any breakpoints. LRR and BAF values were concatenated in each window and added to create a $2 \times n$ feature vector representing one window.

No further information (except breakpoint coordinates extracted at positions of copy number state transition) such as chromosome position was taken into account for window selection. Finally, each feature vector was labeled as ”has breakpoint(s) / positive”, if at least one breakpoint is located within the window or ”has no breakpoint(s) / negative”, otherwise. The training and validation data selection is depicted in Figure 1, (a). The evaluation data set is created by sliding a window with no overlap over the whole sample sequence as shown in Figure 1, (c) (for better presentability only a small subset is shown).

For training, validation and evaluation we chose a window size of $n = 40,000$ probes. (see Section 5.3).

3.2. Cross-Validation Folds

In order to evaluate each model on all available samples, we created a 4-fold cross validation. We divide our 12 samples into 4 separated folds such that each fold consists of 3 samples. In each round of training, the models are trained on 9 samples and tested on the remaining 3.

4. Deep SNP: An End-to-end Neural Network for Breakpoint Detection in SNP Array

The genomic data considered by our approach is characterized by very long sequences of low-dimensionality data which makes analysis with conventional deep learning architectures difficult. In our empirical results we show that state-of-the-art architectures such as VGG (Simonyan & Zisserman, 2014) and DenseNet (Huang et al., 2016) that perform very well in audio (Eghbal-Zadeh et al., 2016; Hershey et al., 2017) and image processing (Simonyan & Zisserman, 2014; Huang et al., 2016) applications are not capable of coping well with these data. Hence, we designed an architecture with specialized units targeting specific processing goals to deal with these challenges. In the following section, we detail the architecture and different units used in Deep SNP.

[1] https://github.com/popitsch/varan-gie

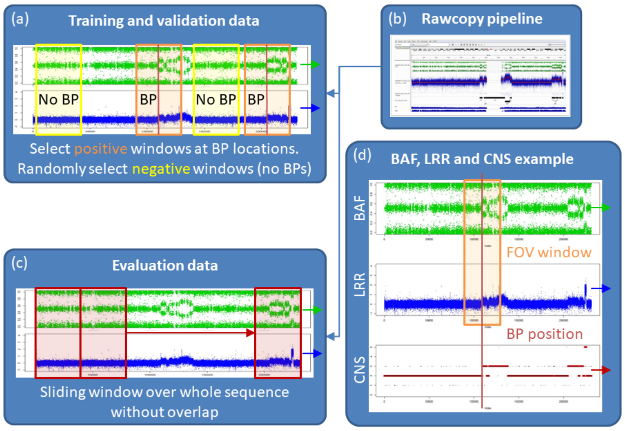
(a) Training and validation data
No BP
BP
No BP
BP
Select positive windows at BP locations.
Randomly select negative windows (no BPs)
(b) Rawcopy pipeline
(d) BAF, LRR and CNS example
BAF
FOV window
LRR
BP position
CNS
(c) Evaluation data
Sliding window over whole sequence without overlap

Figure 1. Data generation pipeline. All images show only a small part of the whole sample sequence with $\sim 225,000$ values. (a) Positive data for training and validation data is created by placing windows at breakpoint (BP) positions. Negative data is created by placing windows at random positions where no BPs are located (e.g. the beginning and/or end of the sequence as well as in between breakpoints. BAF, LRR and truth-set CNS are then extracted in each defined window and concatenated to form the respective input data. (b) Example UI (IGV/VARAN-GIE) of the Rawcopy pipeline, which provides BAF, LRR values and Rawcopy CNS, and enables manual curation. (c) Evaluation data is generated by sliding a window over a full sample sequence ($\sim$ 2.8 Mio values) without overlap (figure shows a subset). Again BAF and LRR values are extracted in each defined window and concatenated. (d) Example window (orange rectangle) placed at a BP position (red line). CNS in the last row depicts the manual truth-set CNS (red horizontal lines) as well as the Rawcopy CNS (black horizontal lines and dots).

4.1. Network Architecture

Deep SNP is a deep neural network trained with stochastic gradient descent (SGD) in an end-to-end fashion. Deep SNP has 5 main units and each unit plays an important role for learning a suitable hidden representation from the genomic data and further learn the relevant aspects of a breakpoint from the long, low-dimensional input sequence.

The Deep SNP units as shown in Figure 2 are as follows: 1) Dilated Convolutional Unit: to cope with the very long SNPa sequences and learn from very low-dimensional data. 2) Feature Learning Unit: To learn a reasonable hidden representation suitable for breakpoint detection. 3) Attention Unit: To be able to focus on specific regions, or use a wide-range of probes. 4) Distributed Recurrent Unit: To learn the dependencies in the learned hidden representation, between hidden factors of different genomic positions, and finally 5) Localization Unit: To be able to decide which of the hidden factors of the genomic positions are more relevant to the task.

Dilated Convolutional Unit Dilated convolutional layers (Yu & Koltun, 2015) are specialized convolutional layers with large receptive fields that can process very long inputs. These layers are known for their significant performance in audio processing and machine translation using long sequences of one-dimensional data, to learn from or generate raw audio/textual input instead of 2D image-like features such as spectrograms. Good examples for successful applications of dilated convolutions in dealing with very long one-dimensional sequences are ByteNet (Kalchbrenner et al., 2016) for fast machine translation, and Wavenet (Van Den Oord et al., 2016) which is the state-of-the-art in speech synthesis and is capable of synthesizing directly very long audio samples, while it is only trained on long sequences of raw audio. Recently, (Gupta & Rush, 2017) their were used to process Genomic data and they have shown great promise.

Feature Learning Unit In this unit, we incorporate and adapt state-of-the-art convolutional architectures which can efficiently learn a high-level representation. We decided to choose a state-of-the-art architecture for image recognition called DenseNet (Huang et al., 2016). Nevertheless, this unit is interchangeable with other architectures such as ResNet (He et al., 2016) or VGG (Simonyan & Zisserman, 2014). We connect the input layer of the feature learning unit to the representation that our dilated convolutions learned. We modified the DenseNet in order to learn from the genomic position dimension which is very long. Therefore, we only used 1D filters and 1D pooling layers.

Attention Unit Biologists often have to spend long hours investigating the probes carefully to be able to annotate a breakpoint as the truth-set. They need to use special tools to explore such long sequences, enabling them to *zoom-in* and *zoom-out* in specific regions of the genomic data to make the final decision about a breakpoint. We use an *Attention Unit* that allows our network to attend on any activations in any desired probe index. This enables Deep SNP to explore the probes and focus on specific regions of the genomic sequence if necessary and process the data in similar ways as the biologist annotators.

Distributed Recurrent Unit Recurrent models such as LSTM (Hochreiter & Schmidhuber, 1997) and GRUs (Chung et al., 2014) are well-known models to model sequential data such as audio, text and genome sequences (Gupta & Rush, 2017). Therefore, we use a bidirectional Gated Recurrent layer (GRU) and apply it on the hidden space that our feature learning module learned. We prefer GRU over LSTM because of its simplicity and competitive performance over LSTM. We apply the GRUs on the sequence of hidden activations formed by the dimension in the hidden space that represents the genomic position of the features. As the filters used are 1D, the dimension in the hidden representation that represents the genomic position can be tracked back to the input and represent its corresponding input probes as it is explained in the localization unit.

Localization Unit Localization Units can provide a mechanism to trace a hidden representation back to the input in a specific dimension. By using distributed units we can repeat a processing unit such as softmax on every node of a specific dimension. This way, the predictions of each node in that dimension represent the corresponding indexes in the input (which directly maps to the respective genomic position).

We tried three different localization mechanisms for localizing the regions in the input probes with high importance for breakpoint prediction. The first, is to apply a distributed dense layer with softmax activations on the dimension of the hidden representation learned from the previous unit on the dimension that represents the genomic position. Then each softmax output can be interpreted as the importance of a series of consecutive probes in the input. For the final prediction, these softmax probabilities are averaged into a final probability and can be used to train the model as there is only one label provided per window. Nevertheless, the output of the distributed softmax before averaging can be used for localization purposes. These results can be found under *No Attention* in Table 1.

For our second localization method, we use a technique from state-of-the-art in audio event detection with weakly-labeled data (Xu et al., 2017). We added an additional attention gate using a distributed dense layer with sigmoid activations that can control the output of the distributed softmax probability by multiplying the output of the sigmoid gate to the softmax prediction. This mechanism allows the network to close the activations related to the genomic position that are not beneficial for the breakpoint predictions. Similar to our first localization unit, the outputs are averaged and used for training. The output of the attention gate can be then used for localization purposes separately. These results can be found under *Final Attention* in Table 1.

For our last localization, we additionally added the attention gate to the input of the recurrent unit. This helped to improve the localization performance, as the inputs can be controlled even before processed by the GRUs. These results can be found under *Mid+Final Attention* in Table 1. Examples of the localization unit using Mid+Final attention are provided in Figure 3.

5. Results

In this section, we provide the empirical results in Table 1 for SNPa breakpoint classification using different methods.

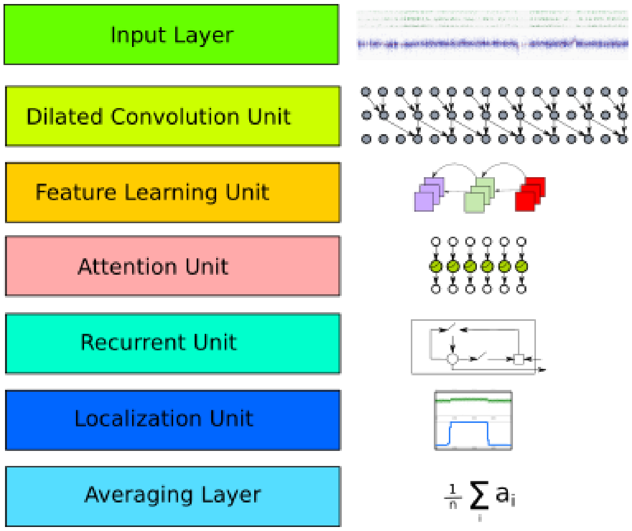

Figure 2. Block-diagram of the proposed Deep SNP.

In addition, we use Deep SNP's localization unit to pinpoint the position of the breakpoint in a window and illustrate breakpoint and genomic segment positions with example views. For these results, *the same* Deep SNP model as used in Table 1 is used which was only trained on binary labels per window and basically had no knowledge about the exact position of the breakpoint(s) within the window. These results are shown in Figure 3.

In Section 5.1 and Section 5.2, we explain the baseline systems and clarify the training procedure in the baselines as well as the proposed method.

5.1. Baselines

This section explains the other network architectures we evaluated on, known as *Baseline methods*. We used two state-of-the-art feed forward deep neural networks namely VGG (Simonyan & Zisserman, 2014), and DenseNet (Huang et al., 2016)) as baseline methods to demonstrate the learning abilities of Deep SNP compared to general purpose neural networks.We also compared our results to the breakpoints detected by Rawcopy, a specialized tool for CNV detection.

5.1.1. Rawcopy

Rawcopy data processing was done according to Section 3. The encoded copy numbers at the probe positions were then evaluated against the manually curated truth-set. Rawcopy uses various built-in reference data precompiled from a large number (2875 samples, Supplementary Table 1 (Mayrhofer et al., 2016)) of ethnically diverse samples, with variations also in technical quality to improve LRR and BAF normalization. Therefore, in the sense of training data, Rawcopy benefited from significantly more data compared to Deep SNP and the rest of the Deep Learning baselines that were only trained with 9 out of 12 available samples.

To evaluate Rawcopy for breakpoint prediction, first the copy number states were predicted for each probe and further converted to our binary format as explained in Section 3.1.

5.1.2. Deep Neural Networks

Feed-forward Neural Networks We used three feed-forward deep neural network architectures that are known to perform reasonable well in various tasks such as image recognition (Huang et al., 2016) and audio acoustic scene classification (Eghbal-Zadeh et al., 2016; Hershey et al., 2017) as baselines. We use two well-known deep convolutional feed forward architectures namely VGG (Simonyan & Zisserman, 2014) used in *BL VGG* and DenseNet (Huang et al., 2016) used in *BL DenseNet*. We adapted the filter sizes of the aforementioned baselines to the task and the data used. Also no dilation, attention, recurrent units or localization units were used in these baselines.

BL Dilated DenseNet is a baseline designed based on the proposed method in (Gupta & Rush, 2017) which uses incremental dilation in a feed-forward CNN to model long-distance genomic dependencies. We adapt their architecture to our task (as their task was not detection) by adding convolutional layers followed by a global average-pooling layer to play the role of a ”classifier” at the final part of the network. We further improve the CNN architecture by upgrading it to a DenseNet architecture with incremental dilation. Further, this design is also aligned with other DenseNet architectures in *BL DenseNet* and the proposed Deep SNP in terms of filter sizes, number of parameters and the overall architecture design. These changes resulted in significant improvements in our task, compared to the initial architecture. The architecture of the implemented *BL VGG*, *BL DenseNet* and *BL Dilated DenseNet* are shown in the appendix (Table 2).

Recurrent Neural Networks Recurrent Neural Networks (RNNs) are known for their ability to model sequential data. Therefore, we use a convolutional bidirectional LSTM (Hochreiter & Schmidhuber, 1997) as our final baseline. We design this baseline based on the results reported in (Gupta & Rush, 2017), as authors replaced incremental dilated convolutions with convolutional layers followed by bi-directional LSTM after the convolutional layers. Similarly, we design our last baseline *BL LSTM DenseNet* based on a model used (Gupta & Rush, 2017). We replace the convolutional layers with a DenseNet architecture and we add a bidirectional LSTM for sequence modeling of the learned representation by the convolutional layers. No dilation, attention, or localization units were used in this baseline. We then add the ”classifier” part of the network similar to the other baselines using a convolutional layer followed by global average-pooling. The architecture of the *BL LSTM DenseNet* is shown in Table 2.

All baselines were trained for 200 epochs with a batch size of 25 using AMSGrad (Reddi et al., 2018) (an adaptive version of SGD), categorical cross-entropy loss, initial learning rate of 0.001 which was reduced on plateau by 0.1. The patience of 50 and early stopping of 100 epochs was carried out as well. Deep SNP as well as deep learning baselines are implemented in Keras (Chollet et al., 2015) and the training is done on a Nvidia DGX Station using Keras with Tensorflow (Abadi et al., 2016) backend.

5.2. Deep SNP Training Strategy

The proposed Deep SNP was trained for 100 epochs with the same hyper-parameters, optimizer and training strategy as the baseline networks (see Section 5.1.2). The architecture of Deep SNP can be found in Table 3.

5.3. Evaluation

All described algorithms were evaluated using 4-fold cross-validation. For Deep SNP and deep learning baselines, in each fold nine samples were used for training and three for validation (selected windows) and evaluation (full sequence). Note that evaluation data was created differently than validation data in terms of the length of data used (see Section 3). For each fold we report F1-Score (F1), Precision (PREC) and Recall (REC). For F1, PREC and REC we apply averaging in two different ways: macro (MAC - calculate metrics for each label, and find their unweighted mean) and binary (POS - calculate metrics only for positive labels i.e. with breakpoint). The window size for training sample selection was $40,000$. This value was chosen based on empirical experiments conducted during algorithm development.

5.4. Empirical Results

The results for each evaluated algorithm are shown in Table 1.

5.5. Extended Empirical Results

In this section we visualize the output of the localization unit in Deep SNP. Please note that these results are produced with the exact same model as used for evaluation and its results are provided in Table 1 under *Mid+Final Attention*. Only instead of the final output predictions, we visualized the output of the localization layer. We need to further clarify that the results of the Section 5.5 are preliminary and need to be further systematically evaluated and compared to the state-of-the-art.

Table 1. Cross-validation results for all conducted experiments. All values are shown in %. Binary precision and recall (PREC POS, REC POS) are highlighted in all experiments.

(%) / Folds	Fold 1	Fold 2	Fold 3	Fold 4
Rawcopy				
F1 Mac	43.42	48.30	23.65	41.49
F1 Pos	17.39	46.60	39.06	37.37
Prec Mac	55.40	48.36	26.76	41.78
Prec Pos	09.71	30.57	24.27	23.12
Rec Mac	68.53	65.75	52.15	63.54
Rec Pos	83.33	97.96	100.00	97.37
BL VGG				
F1 Mac	51.75	22.51	42.12	45.10
F1 Pos	11.43	28.70	0.0	0.0
Prec Mac	85.45	23.0	72.77	82.16
Prec Pos	8.70	18.23	0.0	0.0
Rec Mac	53.11	38.55	47.55	50.0
Rec Pos	16.67	67.35	0.0	0.0
BL DenseNet				
F1 Mac	54.54	43.05	43.20	51.84
F1 Pos	12.50	0.0	0.0	13.64
Prec Mac	93.43	75.59	76.06	82.16
Prec Pos	25.0	0.0	0.0	50.00
Rec Mac	53.42	49.09	49.69	53.09
Rec Pos	8.33	0.0	0.0	7.89
BL Dilated DenseNet				
F1 Mac	70.34	84.29	58.47	76.72
F1 Pos	44.44	74.70	29.03	60.32
Prec Mac	92.96	90.14	79.34	88.26
Prec Pos	40.0	91.18	75.0	76.0
Rec Mac	72.76	80.72	58.08	73.29
Rec Pos	50.0	63.27	18.0	50.0
BL LSTM DenseNet				
F1 Mac	73.77	79.60	77.68	73.27
F1 Pos	50.0	66.67	63.01	53.57
Prec Mac	95.31	87.79	87.32	87.79
Prec Pos	62.50	89.66	**100.0**	83.33
Rec Mac	70.09	75.62	73.0	68.88
Rec Pos	41.67	53.06	**46.0**	39.47
Deep SNP (No attention)				
F1 Mac	90.86	87.27	65.50	86.23
F1 Pos	84.21	80.0	34.78	80.0
Prec Mac	95.71	91.43	92.86	89.05
Prec Pos	96.0	75.0	80.0	79.31
Rec Mac	87.22	89.28	60.85	86.42
Rec Pos	75.0	85.71	22.22	80.70
Deep SNP (Final attention)				
F1 Mac	88.84	93.36	69.39	88.50
F1 Pos	80.70	89.41	42.86	83.18
Prec Mac	94.77	95.71	92.38	90.95
Prec Pos	92.0	88.37	60.0	83.93
Rec Mac	85.37	93.75	65.62	88.28
Rec Pos	71.87	90.48	33.33	82.45
Deep SNP (Mid+final attention)				
F1 Mac	90.58	95.01	74.87	90.82
F1 Pos	83.64	92.13	53.33	86.49
Prec Mac	95.71	96.67	93.33	92.86
Prec Pos	**100.0**	**87.23**	66.67	**88.89**
Rec Mac	85.94	97.02	71.18	90.15
Rec Pos	71.87	97.62	44.44	84.22

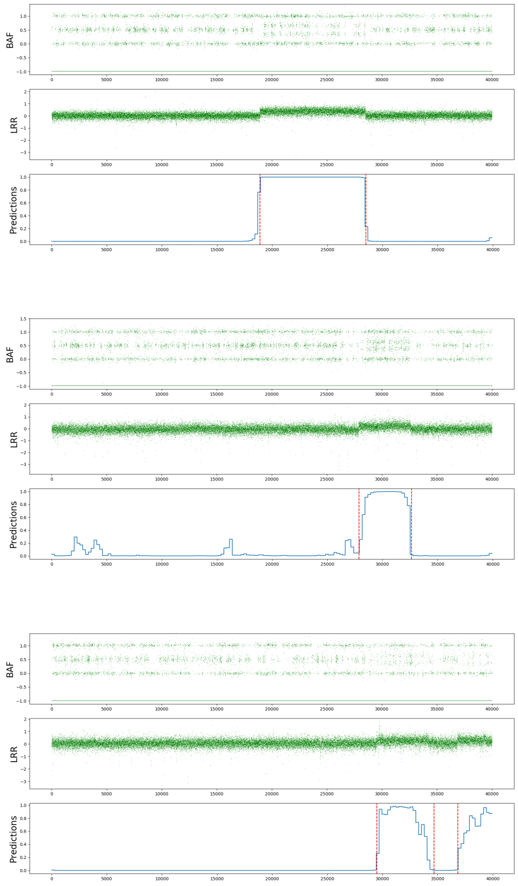

Figure 3. Predictions of the localization unit (with Mid+Final attention mechanism) in Deep SNP. All the 3 windows contain breakpoints. Our model was trained only with whether or not a break point exists in a 40k window. The first row shows the BAF values, the second the LRR values and the third the Deep SNP breakpoint predictions for a given window. The red vertical line represents the truth-set for breakpoints.

5.6. Discussion

As can be seen in Table 1, Deep SNP outperforms BL VGG, BL DenseNet and BL Dilated DenseNet in terms of F1 and Recall. The high precision of BL VGG and BL DenseNet are not meaningful as their recall is very low (15% and 4% for BL VGG and BL DenseNet, respectively) which means only a very small portion of breakpoints were detected by them. BL LSTM DenseNet achieves a better recall (on average 43.13%) but is not still competitive compared to Rawcopy and Deep SNP. In addition we observed that as reported in (Gupta & Rush, 2017), incremental dilated convolution layers can be competitive to LSTMs, while achieving lower precision. On average, BL Dilated DenseNet achieves 43.32% average recall while having average precision of 70.54% compared to average recall of 43.13% and average precision of 83.87%.

In contrast, Deep SNP manages to achieve high average precisions (82.58%, 80.75%, 85.70% for No Att., Fin. Att and Mid.+Fin. Att, respectively) while having reasonable average recalls (65.9%, 69.53%, 74.54% for No Att., Fin. Att and Mid.+Fin. Att, respectively). These results suggests that a right combination of dilated convolution with recurrent layers can achieve significantly better performances as used in Deep SNP.

Rawcopy manages to achieve higher recalls compared to Deep SNP as well as the deep learning baselines. While Rawcopy recall is higher than Deep SNP recall, it achieves lower precisions, mainly because it detects a substantial higher number of false positive segments (the false positive segments are in reality many, and very short).

As can be seen, the results of Fold 3 in all deep learning models are noticeably lower than the other 3 folds. This can be explained by the small training set we used in the current Deep SNP evaluation (9 samples). Interestingly, BL LSTM DenseNet achieves better results on Fold 3 compared to Deep SNP and other baselines, despite its poor performance on other Folds compared to Deep SNP. In contrast, Rawcopy manages to achieve consistent performances in all the folds, which can be explained by the large amount of samples used compared to other methods.

These results suggest the possibility that breakpoints available in the Fold 3 might have been different than the other 3 folds (possibly caused by different copy number state transitions as in the other folds). The performance difference between Rawcopy and Deep SNP can be also explained by comparing the amount of data used in Rawcopy (2.875 samples) and Deep SNP (9 samples). Hence these results could be further improved by providing more training data for Deep SNP.

As can be seen in Figure 3 although Deep SNP was only trained with binary weak labels of breakpoints for the whole windows of 40k probes, its localization unit is not only capable of accurately pinpointing the correct position of a breakpoint, but also the localization predictions stay at a high value during the whole segment of increased/decreased CNS.

6. Conclusion

In this paper, we proposed Deep SNP, a novel Deep Neural Network trained in an end-to-end fashion on SNPa data capable of classifying the presence or absence of one or multiple breakpoints within large genomic windows. We demonstrated the capabilities of Deep SNP by comparing with state-of-the-art architectures as well as a known biological data analysis tool, Rawcopy. Our results showed Deep SNP outperformed other deep models and could achieve performances that are in a reasonable range compared to specialized tools such as Rawcopy. In addition, we showed qualitative examples from the localization unit that can learn to pinpoint the breakpoint positions as well as the genomic segments although these information were not available to the network during training. Therefore, these promising results are a motivation for our future work to investigate the value of DeepSNP for localizing genomic breakpoints and subsequent or simultaneous prediction of genomic segments.

7. Future work

In our future work, we will evaluate the results of the localization unit and will compare it with state-of-the-art methods. Also, we will focus our efforts on improving the recall and also will continue to increase the training data. In addition, we will investigate the possibilities of the use of localization units for segmentation based on copy number state transition. Finally, as we believe Deep SNP is capable of learning from other kinds of genomic data, we will apply Deep SNP on other kinds of genomic data such as WGS data.

Acknowledgements

This work was carried out within the Austrian Research Promotion Agency (FFG) COIN ”Networks” project VISIOMICS and additionally supported by the Austrian Ministry for Transport, Innovation and Technology, the Federal Ministry of Science, Research and Economy, the St. Anna Kinderkrebsforschung and the Province of Upper Austria in the frame of the COMET center SCCH. The authors would like to gratefully acknowledge the support of NVIDIA Corporation with the donation of a Titan X GPU used for this research.

References

Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Ambros, Inge, Brunner, Clemens, Abbasi, Reza., Frech, Christian, and Ambros, Peter F. Ultra-high density snparray in neuroblastoma molecular diagnostics. In *Front. Oncol.*, volume 4, pp. 202, 2014. doi: 10.3389/fonc.2014.00202.

Angermueller, Christof, Pärnamaa, Tanel, Parts, Leopold, and Stegle, Oliver. Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878, 2016. ISSN 1744-4292. doi: 10.15252/msb.20156651.

Benjamini, Yuval and Speed, Terence P. Summarizing and correcting the gc content bias in high-throughput sequencing. *Nucleic Acids Research*, 40(10):e72, 2012.

Ching, T. et al. Opportunities and obstacles for deep learning in biology and medicine. *Journal of the Royal Society Interface*, 15, 2018. doi: 10.1098/rsif.2017.0387.

Chollet, François et al. Keras, 2015.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, and Bengio, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Consortium, The ENCODE Project. An integrated encyclopedia of DNA elements in the human genome. *Nature*, 489(7414):57–74, September 2012. ISSN 1476-4687.

Eghbal-Zadeh, Hamid, Lehner, Bernhard, Dorfer, Matthias, and Widmer, Gerhard. Cp-jku submissions for dcase-2016: A hybrid approach using binaural i-vectors and deep convolutional neural networks. *IEEE AASP Challenge on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

Gupta, Ankit and Rush, Alexander M. Dilated Convolutions for Modeling Long-Distance Genomic Dependencies. *arXiv:1710.01278 [q-bio, stat]*, October 2017. arXiv: 1710.01278.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hershey, Shawn, Chaudhuri, Sourish, Ellis, Daniel PW, et al. Cnn architectures for large-scale audio classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 131–135. IEEE, 2017.

Hochreiter, Sepp and Schmidhuber, Jrgen. Long short-term memory. *Neural computation*, 9(8):1735–80, November 1997. ISSN 08997667 (ISSN).

Huang, Gao, Liu, Zhuang, van der Maaten, Laurens, and Weinberger, Kilian Q. Densely Connected Convolutional Networks. *arXiv:1608.06993 [cs]*, August 2016. arXiv: 1608.06993.

Kalchbrenner, Nal, Espeholt, Lasse, Simonyan, Karen, Oord, Aaron van den, Graves, Alex, and Kavukcuoglu, Koray. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099*, 2016.

LaFramboise, Thomas. Single nucleotide polymorphism arrays: a decade of biological, computational and technological advances. *Nucleic acids research*, 37(13): 4181–4193, 2009.

Marioni, John C, Thorne, Natalie P, and Tavaré, Simon. Biohmm: a heterogeneous hidden markov model for segmenting array cgh data. *Bioinformatics*, 22(9): 1144–1146, 2006.

Mayrhofer, Markus, Viklund, Bjrn, and Isaksson, Anders. Rawcopy: Improved copy number analysis with Affymetrix arrays. *Scientific Reports*, 6:36158, October 2016. ISSN 2045-2322. doi: 10.1038/srep36158.

Olshen, Adam B, Venkatraman, ES, Lucito, Robert, and Wigler, Michael. Circular binary segmentation for the analysis of array-based dna copy number data. *Biostatistics*, 5(4):557–572, 2004.

Pique-Regi, Roger, Monso-Varona, Jordi, Ortega, Antonio, Seeger, Robert C, Triche, Timothy J, and Asgharzadeh, Shahab. Sparse representation and bayesian detection of genome copy number alterations from microarray data. *Bioinformatics*, 24(3):309–318, 2008.

Poplin, Ryan, Newburger, Dan, Dijamco, Jojo, Nguyen, Nam, Loy, Dion, Gross, Sam S., McLean, Cory Y., and DePristo, Mark A. Creating a universal SNP and small indel variant caller with deep neural networks. *bioRxiv*, pp. 092890, 2016. ISSN 1744-4292. doi: 10.1101/092890.

Reddi, Sashank J., Kale, Satyen, and Kumar, Sanjiv. On the Convergence of Adam and Beyond. February 2018.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Van Den Oord, Aaron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Venkatraman, ES and Olshen, Adam B. A faster circular binary segmentation algorithm for the analysis of array cgh data. *Bioinformatics*, 23(6):657–663, 2007.

Wang, Kai, Li, Mingyao, Hadley, Dexter, Liu, Rui, Glessner, Joseph, Grant, Struan FA, Hakonarson, Hakon, and Bucan, Maja. Penncnv: an integrated hidden markov model designed for high-resolution copy number variation detection in whole-genome snp genotyping data. *Genome research*, 17(11):1665–1674, 2007.

Xu, Yong, Kong, Qiuqiang, Wang, Wenwu, and Plumbley, Mark D. Large-scale weakly supervised audio classification using gated convolutional neural network. *arXiv preprint arXiv:1710.00343*, 2017.

Yu, Fisher and Koltun, Vladlen. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Zhang, Zhengdong D and Gerstein, Mark B. Detection of copy number variation from array intensity and sequencing read depth using a stepwise bayesian model. *BMC bioinformatics*, 11(1):539, 2010.

Zhao, Min, Wang, Qingguo, Wang, Quan, Jia, Peilin, and Zhao, Zhongming. Computational tools for copy number variation (cnv) detection using next-generation sequencing data: features and perspectives. *BMC bioinformatics*, 14(11):S1, 2013.

8. Appendix

8.1. Network Architectures

BLVGG
Input $2 \times 40k \times 1$
1×10 Conv(stride-1×3)-32-BN-ReLu 1×5 Conv(stride-1×1)-64-ReLu 1×5 Conv(stride-1×1)-64-BN-ReLu 1×2 Max-Pooling(stride-1×2)+ Drop-Out(0.4)
1×3 Conv(stride-1×1)-128-ReLu 1×3 Conv(stride-1×1)-128-BN-ReLu 1×2 Max-Pooling(stride-1×2)+ Drop-Out(0.3)
1×3 Conv(stride-1×1)-256-ReLu 1×3 Conv(stride-1×1)-256-ReLu 1×3 Conv(stride-1×1)-256-BN-ReLu 1×2 Max-Pooling(stride-1×2)+ Drop-Out(0.2)
1×3 Conv(stride-1×1)-512-ReLu 1×3 Conv(stride-1×1)-512-ReLu 1×3 Conv(stride-1×1)-512-BN-ReLu 1×2 Max-Pooling(stride-1×2)+ Drop-Out(0.2)
1×3 Conv(stride-1×1)-512-ReLu 1×3 Conv(stride-1×1)-512-ReLu 1×3 Conv(stride-1×1)-512-BN-ReLu 1×2 Max-Pooling(stride-1×2)+ Drop-Out(0.2)
1×1 Conv(stride-1×1)-2-ReLu Global-Average-Pooling
2-way Soft-Max

BL DenseNet
Input $2 \times 40k \times 1$
1×10 ConvBn(stride-1×5)-208 + DO
Feature Learning Layers (Table 4)
1×1 Conv(stride-1×1)-2-ReLu Global-Average-Pooling
2-way Soft-Max
BL Dilated DenseNet
Input $2 \times 40k \times 1$
1×10 DilConvBn(dilation-1×3)-208 + DO
Feature Learning Layers with dilation (Table 4)
1×1 Conv(stride-1×1)-2-ReLu Global-Average-Pooling
2-way Soft-Max
BL LSTM DenseNet
Input $2 \times 40k \times 1$
1×10 ConvBn(stride-1×5)-208 + DO
Feature Learning Layers without dilation(Table 4)
Bi-directional LSTM-256
1×1 Conv(stride-1×1)-2-ReLu Global-Average-Pooling
2-way Soft-Max

Table 2. Architecture of the deep learning baselines. BN = Batch Normalization, ReLu = rectified linear unit, Conv = 2d convolution layer. Each layer consists of the filter size, layer type, stride size, number of filters and optional layers e.g. 1×10 Conv(stride-1×3)-32-BN-ReLu describes a 2D convolution layer with filter size = 1×10, stride size = 1×3, 32 filters, batch normalization and ReLu activation.

Input $2 \times 40k \times 1$	
1×10 DilConvBn(dilation-1×5)-208 + DO	
Feature Learning Layers (Table 4)	
Mid-Attention Layer	No-Attention
Bi-directional GRU-256	
Attention Layer	Softmax
Averaging Layer	

Table 3. Architecture of Deep SNP. BN = Batch Normalization, ReLu = rectified linear unit, Conv = 2d convolution layer, ConvBn = Conv+BN+ReLu, DO = Drop-Out(0.2)
Each layer consists of the filter size, layer type, stride size, number of filters and optional layers e.g. $4 \times [1 \times 3$ ConvBn(stride-1×1)-12 + DO $]$-64 describes a DenseBlock containing $4\times$ 2D convolution layer with filter size = 1×3, stride size = 1×1, 12 filters, batch normalization and ReLu activation, Drop-Out(0.2) and a final filter size of 64.

	Deep SNP Feature Learning\BL Dilation DenseNet	BLDenseNet
Input	$2 \times 40k \times 1$	
Convolution	[1 × 5 DilConv dil.: 1 × 2\1 × 9]-16	[1 × 5 Conv pad: same stride:1 × 2]-16
Batch-Normalization	Batch-Normalization	
Dense Block (1)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	
Transition Layer (1)	[1 × 5 DilConv dil.:1 × 2, \1 × 27]-64	[1 × 5 Conv pad: same stride 1 × 1]-64
	1 × 2 average pool dil. 1 × 3, stride 1 × 2	
	Batch-Normalization	
Dense Block (2)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	
Transition Layer (2)	[1 × 5 DilConv dil. 1 × 2, \1 × 81]-112	[1 × 5 Conv pad: same stride: 1 × 2]-112
	1 × 2 average pool stride 1 × 2	
	Batch-Normalization	
Dense Block (3)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	
Transition Layer (3)	[1 × 2 Conv stride 1 × 2]-160	
	1 × 2 average pool stride 2	
	Batch-Normalization	
Dense Block (4)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	
Transition Layer (4)	[1 × 2 Conv, stride 1 × 2]-208	
	1 × 2 average pool stride 2	
	Batch-Normalization	
Dense Block (5)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	
Transition Layer (5)	[1 × 2 Conv stride 1 × 2]-256	
	1 × 2 average pool stride 2	
	Batch-Normalization	
Dense Block (6)	$\begin{bmatrix} [1 \times 3 \text{ Conv}]\text{-12} \\ \text{concat input} \end{bmatrix} \times 4$	

Table 4. Feature learning architectures for Deep SNP and BLDenseNet.