De-MISTED: Image-based classification of erroneous multiple sequence alignments using convolutional neural networks

Hiba Khodji, Pierre Collet, Julie D. Thompson, Anne Jeannin-Girardon

De-MISTED: Image-based Classification of erroneous Multiple Sequence Alignments using Convolutional Neural Networks

Hiba Khodji, Pierre Collet, Julie D. Thompson, Anne Jeannin-Girardon

Abstract—The widespread use of high throughput genome sequencing technologies has resulted in a significant increase in the number of available sequences, creating new challenges for genome annotation and prediction of protein-coding genes, in terms of error detection and quality control. Multiple Sequence Alignments (MSA) of the predicted protein sequences provide important contextual information that can be used to distinguish errors (caused by artifacts in the raw genome data, badly predicted gene sequences, or the alignment methods themselves) from true biological events, either by human expertise or statistical analysis of the sequence data. Here, we propose a new approach that consists in using visual representations of MSAs from an in-house dataset, in which errors are carefully identified, as inputs of Convolutional Neural Networks (CNN) classifying MSAs into erroneous and non-erroneous categories. Our model, called De-MISTED (Deep learning for MultIple Sequence alignmenTs Error Detection) shows a high accuracy (87%) and sensitivity (92%) in identifying MSAs containing erroneous sequences. Visual explanation techniques show that our model correctly identifies the correct position of multiple errors of different types (insertions, deletions and mismatches). Close examination of the data showed that our model can also correctly identify errors that were not annotated in the data. The De-MISTED method thus contributes to a more robust exploitation of the genome data.

Index Terms—Multiple Sequence Alignment, Error detection, Visual Recognition, Convolutional Neural Networks, Binary Classification.

1 Introduction

Multiple Sequence Alignment (MSA) is a widely used technique in the fields of molecular biology, computational biology, and bioinformatics [1]. MSA consists in arranging biological sequences (DNA, RNA or proteins) in a matrix in order to identify regions of similarity and divergence reflecting their biological relationships [2]. It is used, for example, to perform phylogenetic studies of related organisms and to predict molecular structures and functions [3]. MSAs thus provide useful information, which underlines the importance of their accuracy [4]. However, generating MSAs is a computationally challenging task; consequently, errors often occur which are detrimental to the subsequent applications [5]. Errors in MSAs can result from two main sources. First, MSA algorithms can cause misalignments where similar sequences are not correctly identified [4]. Second, the sequences themselves can also contain errors. For example, in the case of protein sequences, sequence misprediction errors are introduced by algorithms used to predict protein-coding genes in DNA sequences. These algorithms are not always accurate and have thus led to a certain amount of inconsistencies in today's protein databases [6], [7].

Although there exists a number of algorithms designed specifically to identify inconsistencies in MSAs [8] [9] [10], to the best of our knowledge, there are no studies introducing an image-based approach to solve this problem. In this paper, we propose a novel method, called **De-MISTED** (**De**ep Learning for **M**ult**I**ple **S**equence alignmen**T**s **E**rror **D**etection), that consists in using images to detect inaccurate MSAs of protein sequences.

Convolutional Neural Networks (CNN) are one of the most extensively used neural networks in Deep Learning. CNNs are particularly praised for their ability to learn hierarchical representations from data and extract relevant information from images for the task at hand. This ability has led to high performances in image classification tasks applied to a diverse and wide spectrum of fields, ranging from satellite to medical imagery. Binary classification consists in classifying instances into one of two classes; it is a supervised learning problem, since a labelled dataset is used to teach the model about the different classes. In the field of bioinformatics, analyzing a group of large multiple sequence alignments in search of errors can be a laborious and strenuous task. In order to overcome this issue, this paper proposes a straightforward binary image classifier, based on CNNs, to filter out MSAs of protein sequences that contain gene-prediction/sequence errors from an in-house built image dataset.

This paper is organized as follows: Section 2 introduces related works and a description of our classification problem. Our dataset is presented in Section 3. Section 4 describes the experimental protocol used for the proposed approach. Sections 5 and 6 discuss the conducted experiments and their results. Finally, Section 7 concludes this paper and draws some perspectives for future works.

• *H. Khodji, P. Collet, J. D. Thompson and A. Jeannin-Girardon are with the ICube Laboratory, UMR7357, University of Strasbourg, 1 rue Eugène Boeckel, 67000 Strasbourg, France*
E-mail: jeanningirardon@unistra.fr

Manuscript received April 19, 2005; revised August 26, 2015.

2 Related works

2.1 Multiple Sequence Alignments

Multiple Sequence Alignment (MSA) is the first step in analyzing and solving many bioinformatics-related problems [1] such as protein structure and function annotation, evolutionary studies, analysis of effects of genetic mutations, *etc.* It is, therefore, considered as one of the most studied problems in computational biology. The MSA problem consists in finding an optimal alignment of three or more biological sequences. Aligning thousands of sequences and producing high-quality alignments require the use of advanced and highly sophisticated methods. To this end, a wide range of algorithms has been proposed over the years, as reviewed in [3], including dynamic programming, progressive multiple alignment, iterative alignment, Hidden Markov Models, and Genetic Algorithms, among others. By default, MSA algorithms create alignments in a simple text format, such as FASTA [11]. The FASTA output, albeit simple, is limited to the sequence data and does not allow the inclusion of annotation or meta-data. To overcome this, the authors in [12] introduced ADOMA (Alternative Display Of Multiple Alignment) that proposes a graphical interface with either a simplified multiple alignment output, where only the mismatches with the parent sequence are shown, or a colored output where amino acids (for protein sequences) with similar physico-chemical properties share the same color. Figure 1 shows an alignment of parts of protein sequences produced by ADOMA in a simplified and colored format.

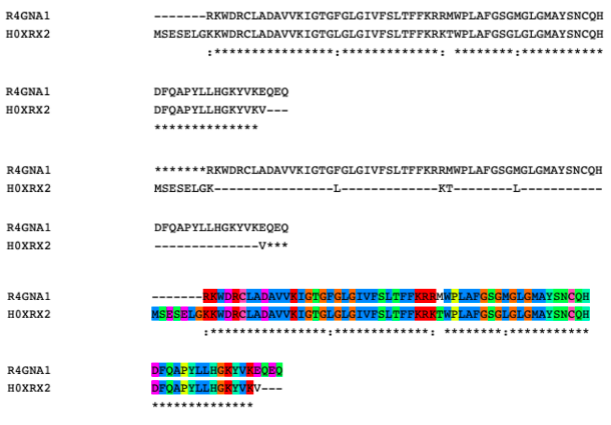


Fig. 1. Example of protein sequences shown in ADOMA's simplified and colored output format. The upper alignments are the default output format. The middle and bottom alignments represent the simplified and colored output formats, respectively.

The MSAs used in this paper contain protein sequences, which are predicted from sequenced genomes. In these alignments, three or more *related* protein sequences are aligned in order to identify regions of similarity. The main goal of an MSA is to match residues (amino acids) from the aligned sequences. The resulting alignment is a rectangular array where protein sequences are arranged *horizontally*, and residues are matched *vertically* so that residues in a given column are homologous, structurally superposable, and/or share a functional role [13]. In other words, *both the vertical and horizontal contexts of an MSA are important to study and analyze it*.

Errors in an MSA can result either from (i) DNA sequencing errors, (ii) errors in the prediction of the exon/intron structure of the protein-coding genes, (iii) misalignment errors in the MSA construction process. These errors include the presence of unusual deletions, insertions/extensions, or mismatches that are inconsistent with the local context surrounding them. A deletion error refers to one or more absent amino acids from a sequence. An insertion involves the inclusion of an additional sequence segment between two amino acids, while the term "extension" is used specifically when an additional sequence segment is added on either end (N- or C-terminal) of a sequence. Finally, a mismatch is represented by non-homologous amino acids in one or more columns of the alignment.

Here, we refer to alignments containing at least one of the aforementioned errors as *erroneous*, while *error-free* MSAs denote accurate alignments with no known inconsistencies.

2.2 Identifying errors in MSAs

A wide range of tools have been proposed in order to characterize misalignment errors. In [14], the authors proposed a knowledge-based approach, called RASCAL, which is used to refine, correct, and improve either automatic or manually constructed multiple sequence alignments. The RASCAL algorithm involves two stages: first, it performs an alignment scanning and validation, which consists in localizing well aligned regions in a given alignment. Second, it detects potential misalignments and performs a re-alignment using dynamic programming. This method was evaluated using MSAs from three different databases: a total of 142 high-quality multiple alignments from the BAliBASE benchmark database [15] were constructed using two progressive programs, ClustalW [16] and MAFFT-2 (FFT-NS2) [17], one iterative technique, MAFFT-I (FFT-NSI) [17], and one cooperative program T-COFFEE [18]. These alignments were then refined using RASCAL. The quality of the produced alignments was assessed using two different scores: the *SP* (sum-of-pairs) score, which indicates the percentage of correctly aligned pairs of residues, and the *Column* score which represents the percentage of correctly aligned columns in the alignment [14]. Using these scores, the authors reported significant improvements for the alignments produced by RASCAL compared to the input MSAs. A subset of 946 alignments from the ProDom protein domain database [19] was also used to evaluate the accuracy and reliability of RASCAL. The alignments were refined using RASCAL and evaluated using the NorMD score [20]. The authors observed an increase in the NorMD score for 68% of the alignments after refinement by RASCAL. A final evaluation step consisted in using a set of 695 full-length nuclear receptor proteins aligned using the MAFFT-2 program. This alignment was then refined using RASCAL and MAFFT-I. From an initial value of 0.27 (by MAFFT-2), the NorMD score was increased to 0.57 after refinement by RASCAL, and to 0.44 after using the iterative program MAFFT-I. In addition, RASCAL yielded an increase in alignment accuracy using less CPU time compared to MAFFT-I: while RASCAL required 9 minutes to detect and realign the errors, MAFFT-I took 4.5 hours. Other knowledge-based approaches include the integration of other information, such as structural elements [21] or evolutionary mutation rates [22].

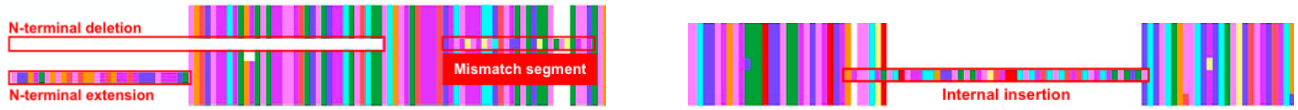
In order to identify errors in MSAs due to inaccurate prediction of protein-coding genes, Khenoussi et al. [8] developed a Bayesien model named SIBIS, which takes an MSA as input and highlights all identified inconsistencies in an XML file. In order to evaluate the proposed approach, SIBIS was applied to protein sequences predicted from the *rhesus macaque* genome [23]. The obtained results were compared with a reference set of 90 protein sequences with experimentally validated errors. The authors assessed the performance of SIBIS against that of two different algorithms: MisPred [24] [25] and their previously developed method [26]. The accuracy of SIBIS was estimated in terms of sensitivity and specificity; the reported results showed a higher sensitivity for SIBIS (81%) compared to 27% for MisPred and 62% for the profile-based developed method. The authors observed that while some errors were overlooked by SIBIS, they were correctly identified by MisPred, and therefore considered the two methods to be complementary. Moreover, SIBIS reached 92% in specificity while the profile-based method achieved 96%. However, the slight loss in specificity by SIBIS was considered statistically insignificant. Other knowledge-based methods have also been developed to address this issue, including FixPred [27] that identifies mispredicted sequences using information about known functional regions. In [28], Jehl et al. proposed a software, called OD-seq, which detects outlier sequences in a given MSA. The algorithm uses a simple gap-based distance metric to find outliers with unusual average distances to the rest of the sequences. These distances are found in two ways: bootstrapping or interquartile range analysis. The performance of OD-seq was evaluated on over 1000 Pfam families [29], where sequences were seeded with outliers either from the same or different family and realigned using Clustal Omega. OD-Seq was also tested on unaligned sequences. When outliers were introduced from a different family, OD-Seq is faced with clear outliers and therefore achieved a good performance with area under the curve (AUC) values of 0.978, 0.96, and 0.936 for aligned bootstrap (Ab), interquartile range (IR), and unaligned bootstrap (Ub), respectively. When outliers are seeded from the same family, however; the difference between an outlier and a homologous is not so obvious which led to a drop in performance in AUCs (Ab= 0.759, IR= 0.745, Ub=0.732). The authors also used Precision-recall curves and inferred that bootstrap analysis yielded a more robust prediction than interquartile range.

In a similar line of research, Chiner-Oms et al. [30] developed EvalMSA, a software tool for evaluating and identifying divergent sequences or outliers in MSAs. EvalMSA analyzes the length of the sequences used to construct the alignment and assigns a weight (*i.e.* influence on the alignment quality) to each sequence; the program then computes the "gappiness" (gpp) value for each sequence; sequences with the highest gpp value are considered strongly divergent and more likely to introduce gaps in the remaining sequences. The authors explored the results of their approach using an MSA from the Pfam database [29]. Three outliers were added to the alignment which was then realigned with MUSCLE [31]. EvalMSA identified two outliers based on their high gpp (gappiness) values, while the third outlier had the lowest weight amongst all sequences. The authors compared their approach to OD-Seq [28] on the same MSA and observed that while EvalMSA identified all three outliers within the alignment, OD-seq identified only those with the highest gappiness values and maintained the third in the core alignment. While both OD-Seq and EvalMSA use the number of gaps between sequences, EvalMSA also takes into account gpp values as well as weights to identify outliers and their cause of divergence. The authors argue that EvalMSA provides a deeper analysis and can be complementary to OD-seq.

2.3 Deep Neural Networks and MSA

Deep Neural Networks are widely used in many aspects of scientific research to find solutions to complex problems. In bioinformatics, experts have also benefited from the analytical ability of DNNs to extract insights from data. In [32], the authors have proposed a deep learning solution to the problem of Multiple Sequence Alignment (MSA) construction by using Deep Reinforcement Learning (DRL). Their proposed approach utilizes Q-learning and LSTM —Long Short-Term Memory networks— for their ability to memorize events. The proposed DRL technique was compared with some of the most popular tools for solving the MSA problem: ClustalW [33] and MAFFT [34], and tested on the *Lemur, gorilla and Mouse* (LGM) and *Rat, Lemur and Opossum* (RLO) datasets [35]. While the DRL approach, ClustalW, and MAFFT achieved an equal SP score on the LGM dataset, the authors demonstrated that their approach achieved a better performance compared to its matrix-based counterparts on the Column Score, Alignment Length, and Exact Match scores. On the RLO dataset, the DRL approach scored 486 on the SP score, which is higher than 480 for ClustalW and 471 for MAFFT. The proposed approach was also compared with a reinforcement learning (RL) technique, proposed in [36], and tested on six datasets: LGM, RLO, *Papio Anubis* (PA) and *Hepatitis C virus* (HCV) from the EMBL nucleotide sequence database [37], dataset 469 (d469) and dataset 429 (d429) from *oxbench_mdsa_all* [38]. The results showed that the DRL and RL approaches scored equally on the SP score on d469, RLO, LGM, and HCV. On the other hand, the proposed DRL approach achieved better results compared to the conventional RL technique on both PA and d429: (PA: 18 860; d429: 10 218) for the DRL approach and (PA: 18 719; d429: 86 68) for the RL technique. In terms of computational time, the DRL approach was tested on two other datasets against a different RL technique [39]. The proposed approach proved to have a faster convergence: 486 s against 1334 s for the RL technique on the first dataset, and 35 s against 101 s on the second dataset.

In other lines of research, Zhang et al. [40] developed an open source deep learning-based program, called DeepMSA, for sensitive MSA construction. This method merges sequences from whole-genome and metagenome sequence databases. In [41], Aoki et al. benefited from deep learning technology by proposing a one-dimensional CNN method to classify pairwise alignments of ncRNA (non-coding RNA) sequences as either related or unrelated. The proposed CNN approach takes as input a distributed representation of a pairwise alignment with gaps of two sequences; each column represents a pair of five-dimensional



(a) MSA with three different types of errors: a deletion, an extension, and a mismatch.

(b) MSA with an insertion error.

Fig. 2. Manually annotated alignments from our in-house built image dataset.

vectors, for the four nucleotides of RNA and the gap symbol, as well as a three-dimensional vector for representing secondary-structure information specific to ncRNAs.

2.4 Convolutional Neural Networks

Convolutional Neural Networks are a type of deep neural networks which proved to be more efficient than regular neural networks for image processing systems. AlexNet [42], VGG [43], GoogLeNet [44], Inception [45], and ResNet [46] are popular CNN models, which have achieved state-of-the-art performances in image classification. While the best performance in ILSVRC-2010 (ImageNet Large-Scale Visual Recognition Challenge) [47] achieved a top-1 error of 47.1% and a top-5 error of 28.2%, AlexNet [42] yielded top-1 and top-5 test set error rates of 37.5% and 17.0%. In the ILSVRC-2012 competition, a variant of AlexNet yielded a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry. ResNet [46] has won first place in the ILSVRC-2015 competition with an ensemble of six models with different depths which led to 3.57% top-5 error on the test set.

In medical imaging, CNNs' performance was compared with that of human experts on image classification tasks: the authors in [48] have developed a 121-layer convolutional neural network, called CheXNet, to detect pneumonia from chest X-rays. The authors evaluated the performance of their model and found that CheXNet yielded a higher F1 score (0.435) compared to average radiologist performance (0.387).

CNNs' rising popularity is attributed to their ability to extract hierarchical features from data. For instance, in image classification, it has been shown that low level layers in a CNN learn general features (colors, simple shapes, textures, *etc.*), while high level layers can learn more abstract representations [49].

In this work, CNNs were used to process and dissect images of multiple sequence alignments in order to detect inconsistencies in given image MSAs. The detection of erroneous multiple sequence alignments is a binary classification problem, where the input is an image of a multiple sequence alignment predicted to belong to either the class `NO_ERROR` or the class `ERROR`, indicating the absence or presence of *at least one error* in a given MSA, respectively.

3 Dataset

3.1 Description

The dataset used in this paper consists of 19 942 multiple sequence alignments from a set of automatically annotated alignments described in [7]. In [7], multiple sequence alignments were used to detect sequence errors in protein sequences, extracted from the Uniprot reference proteomes and RefSeq databases, using the SIBIS algorithm [8]. Of all the 19 942 annotated multiple sequence alignments, 12 545 were identified as containing errors and the remaining 7 397 alignments were found to be error-free. As described in Section 2.1, three types of errors can be found in an alignment: deletions, insertions/extensions, and mismatches. The authors of [7] were able to detect 44 001 deletions, 27 289 insertions, and 11 015 mismatches. It is important to note that, since the original alignments weren't annotated manually, some alignments may contain certain errors that were overlooked by the annotation tool.

In order to obtain a balanced dataset, we used 3 811 alignments from the set of 12 545 erroneous MSAs from which we removed the erroneous sequences using a filtering protocol [1] in order to produce 3 811 *error-free* alignments. The original 3 811 *erroneous* MSAs were conserved, and in total we obtained 11 208 error-free alignments and 12 545 alignments containing errors.

The next step in building our dataset is to convert the alignments from their XML format into images, using the ADOMA [12] command line tool. ADOMA takes a FASTA file as input and produces either a simplified multiple alignment output, or a colored output where amino acids with similar physico-chemical properties share the same color. For our study, we were only interested in the colored output, which we slightly adjusted to remove the characters and their color code to make the sequences more contrasted. XML format files were converted to FASTA format using ClustalW [16] and then passed as input to ADOMA to produce HTML files in the colored format. Finally, the HTML files were converted into images using an open source command line tool [50]. Figure 2 shows two alignments with annotated errors from our created image dataset. The errors were manually highlighted with bounding boxes. Three different errors are shown in alignment (a): a deletion, an extension, and a mismatch error. Alignment (b) shows an internal insertion error.

During the final conversion process, from HTML format to JPG, some alignments were too large for the ADOMA program, and therefore resulted in 419 blank images. After filtering out these blank images from the entire dataset, we obtained a total of 23 334 images. These images are then used as input to a binary classifier in order to be categorized into one of two classes: `NO_ERRORS`: for error-free MSAs, and `ERRORS`: for MSAs containing at least one error.

1. The filtering protocol is a simple program that takes as input an erroneous MSA in XML format and filters out erroneous sequences which are defined by specific *start* and *end* tags

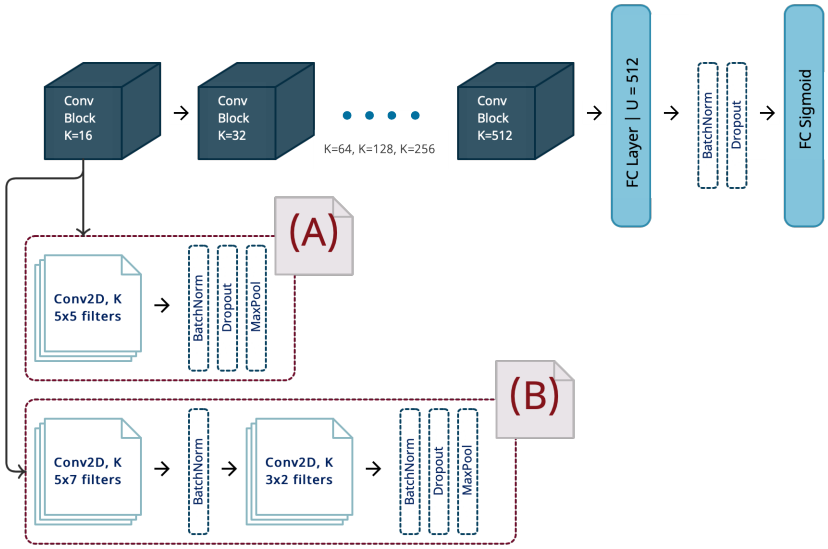


Fig. 3. Models (A) and (B) share the same backbone: six convolutional blocks followed by a fully connected layer and a sigmoid layer. Each convolutional block consists in a single convolutional layer for model (A), and two convolutional layers for model (B). While the filters of (A) are square, the filters of model (B) are rectangular, in order to consider the horizontal and vertical contexts of a sequence alignment.

3.2 Data split

In order to train our classifiers, we split our data into 60%, 20%, and 20% for our training, validation, and test sets, respectively. Figure 4 summarizes the distribution of samples between two classes: ERRORS (*i.e.* sequences containing at least one error) and NO_ERRORS (*i.e.* sequences containing no errors) across our training, validation, and test subsets. There are no duplicates between the subsets. Each MSA contains the primate orthologs of a different human protein. Although there could be some redundancy at the sequence level due to paralogous sequences, each MSA contains a unique human sequence and each MSA is therefore different.

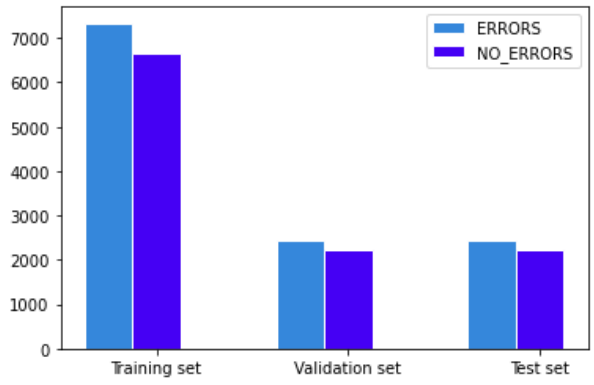


Fig. 4. Distribution of samples across training, validation, and test sets.

4 Experimental Protocol

We defined two convolutional neural network architectures, which are depicted in Figure 3. Both architectures (A) and (B) share the same backbone: six convolutional blocks computing respectively 16, 32, 64, 128, 256, and 512 feature maps, followed by a fully connected layer with 512 units, and a final sigmoid layer computes the output of our binary classification. For Model (A), we defined a single convolutional layer for each block. In model (B), each convolutional block contains two convolutional layers. Since the images in our dataset are of a large rectangular shape, they were downsampled to 224x1024 for both models.

The key difference between the proposed models is the size of the convolutional filters. In model (A), we used 5x5 filters in each convolutional layer. For model (B) we considered the following: as described in Section 2.1, it is important to take into account the structural shape of a MSA: while the constituting protein sequences are arranged in a horizontal manner, homologous amino acids of each sequence are matched vertically to highlight the conserved regions. Therefore, given the horizontal nature of the protein sequences and the vertical context of the alignment, we used a *horizontal* filter (5x7) in the first convolutional layer; and a *vertical* filter (3x2) in the second: an approach which we refer to as **hybrid filtering**.

The models were trained during 100 epochs using a batch size of 32. The training process was optimized using the Adam optimizer (learning rate: $1e^{-2}$, weight decay: $1e^{-4}$) using a binary cross-entropy loss. In order to improve the speed and performance of our network, Batch Normalization (BN) is added after each convolutional layer, resulting in a significant decrease in training time.

A deep neural network with a large number of parameters is prone to overfitting [51]. This can be avoided using a regularization technique. The most commonly used regularization technique is the Dropout method, which we added after each batch-normalization layer. Early stopping was used with a patience of 10 training epochs. The initial learning rate starts at $1e^{-2}$, and is reduced after 5 epochs with a factor of 0.1 if the validation loss does not improve.

5 Experimental Results

In order to assess the performance of our binary classifiers, we plotted their training and validation losses and accura-

cies over the training epochs. For model (A) the training stopped after 25 epochs and reached a validation accuracy of around 86%, while the training and validation losses decreased to around 0.3. Model (B) stopped training after 30 epochs, while achieving the same validation accuracy ($\sim 86\%$) and loss (0.3) as model (A).

We tested our binary classifiers on the test set and evaluated their performances; while both models yielded an accuracy of **87%**, they achieved slightly different loss values: **0.34** for model (A), and **0.32** for model (B). However, the classification accuracy and loss metrics are not conclusive assessment tools; therefore, we used the F1-Score to measure the class-identification ability of our trained models. The F1-score combines two metrics: the *precision*, which quantifies the number of correctly predicted instances over all positive predictions, and *recall*, which measures the number of correct positive predictions out of all positive instances in the dataset. Our aim is to evaluate our proposed models on their ability to identify as many erroneous MSAs as possible with a high precision; in other words, we are interested in finding the best optimal combination of precision and recall for the ERRORS class. Table 1 provides an overview of model (A)'s precision, recall, F1 score, accuracy, and loss for each class. Figure 5 shows the confusion matrix of this model, and provides the number of correctly and incorrectly classified instances.

TABLE 1
Classification report for model (A)

Class	Precision	Recall	F1 Score	Acc.	Loss
ERRORS	0.84	0.92	0.88	0.87	0.34
NO_ERRORS	0.90	0.80	0.85		

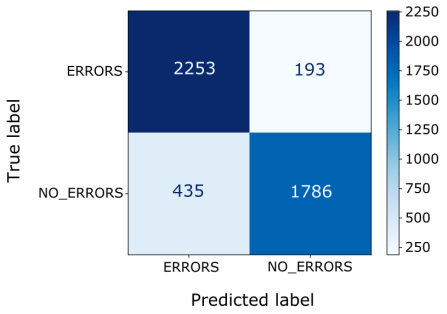


Fig. 5. Confusion Matrix for model (A)

Table 2 and Figure 6 depict the classification report and confusion matrix for model (B), respectively. From Tables 1 and 2, we observe that both models achieved the same F1 Score (0.88) with a slight drop of 1% in the NO_ERRORS class for model (A). While both models achieved the same recall score of 0.92 for the ERRORS class, model (B) reached a precision score of 0.85 compared to 0.84 by model (A). **This means that model (B) correctly identifies 92% of all erroneous MSAs in the test set with a precision of 85%.**

TABLE 2
Classification report for model (B)

Class	Precision	Recall	F1 Score	Acc.	Loss
ERRORS	0.85	0.92	0.88	0.87	0.32
NO_ERRORS	0.91	0.82	0.86		

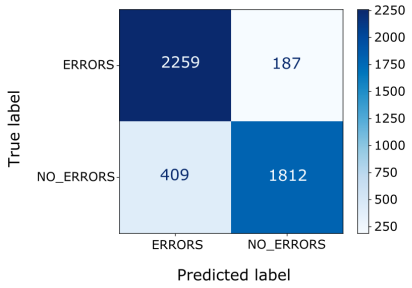


Fig. 6. Confusion Matrix for model (B)

In order to better interpret the resulting confusion matrices in Figures 5 and 6, it is important to remember that the original MSAs used to create the dataset were annotated using SIBIS, an automatic Bayesian model [8]. According to the results reported in [8], SIBIS yielded better specificity (92%) than sensitivity (81%), which means it is possible that some errors within the original dataset have been left undetected by SIBIS. This indicates that a higher number of False Negatives (error-free alignments classified as erroneous) is tolerable, since it is very likely for an initially error-free MSA (as annotated by SIBIS) to be identified as containing at least one error by our models.

As shown in Figures 5 and 6, while model (A) misclassified 193 (7.9%) erroneous MSAs, model (B) misidentified slightly less erroneous MSAs, 187, (7.6%) as error-free. On the other hand, both models found an important number of initially error-free set of alignments to contain at least one error: 435 (19.6%) for model (A), and 409 (18.4%) for model (B).

Classifying MSAs as containing at least one error or error free relies on the ability of our models to identify different types of errors within MSAs. To assess this ability, we compared the number of all errors present in the test set with the number of errors found in the correctly classified erroneous alignments by models (A) and (B). It is important to note that, in order to conduct this comparison, we used the annotation tool, SIBIS, on the original XML version of the MSAs to identify the type of errors found within.

Deletion errors are the most present in the test set, with a total of 5 158. Mismatches were found to be the least present, with only 419 occurrences. From 6 793 errors (of all types) in the test set, 6 485 errors (95.5%) were found across all correctly predicted erroneous MSAs by model (A), while **6 498** (95.6%) were found in the correct erroneous predictions by model (B). Considering that MSAs can contain multiple errors of different types, it is difficult to determine which errors were identified by the models that led to a correct classification. To this end, we also compared the number of errors in the correctly classified *single-error* MSAs by models (A) and (B) and reported the results in Table 3. Out of 825 errors in the test set, model (A) identified

TABLE 3
Comparative table of the number of errors (per type) present in the test set and the number of errors found in the correctly classified erroneous predictions of models (A) and (B). We consider all MSAs as well as MSAs containing only a single error.

	Number of errors in the test set		Number of errors in TN (A)		Number of errors in TN (B)	
Error type	all MSAs	single-error MSAs	all MSAs	single-error MSAs	all MSAs	single error MSAs
N-terminal extension	436	89	394	71	403	73
N-terminal deletion	2 058	298	1 991	278	1 986	282
C-terminal extension	179	23	159	16	157	16
C-terminal deletion	619	96	582	80	591	84
Internal insertion	601	59	559	32	555	34
Internal deletion	2 481	227	2 416	200	2 422	196
Mismatch	419	33	384	21	384	17
Total	6 793	825	6 485	698	6 498	702

TN (True Negatives) : the correctly classified erroneous MSAs.

698 errors (84.6%), while model (B) found **702** (85%). Across all different error types (with the exception of the mismatch and internal deletion types), model (B) achieves a relatively higher number of identified errors than model (A): 87 extensions (N- and C- terminal) were found by model (A) (77.7%) compared to **89** (79.5%) by model (B); 358 deletions (N- and C- terminal) were detected by model (A) (90.9%) compared to **366** (92.9%) by model (B); lastly, 32 insertions were identified by model (A) (54.2%), while **34** (57.6%) were found by model (B). These comparative results further support our previous conclusion that model (B) exhibits a slightly better predictive performance than model (A).

Overall, both models (A) and (B) exhibit good performance when distinguishing between erroneous and non-erroneous MSAs, regardless of the kind of error (although insertions are more challenging to identify). This supports our idea that visual representations of MSAs contain useful information that Convolutional Neural Networks can use.

5.1 Performance comparison

We compared the performance of our proposed CNN-based approach against EvalMSA [30] and OD-seq [28]. Both algorithms are based on the concept of gap penalty; OD-Seq uses an alignment gap metric to measure gap-based distances between sequences in a pairwise fashion, and identifies sequences with missing parts compared to the rest of the alignment. EvalMSA computes a gappiness value (*gpp*) for each individual sequence in order to measure its contribution to insert gaps in the alignment; sequences with high gpp values are considered divergent from the rest. We evaluated these methods on our test data using precision and recall metrics for each class and reported the results in Table 4. It is important to note that these tools do not perform a binary classification of MSAs; their goal is to detect outlier sequences within a given alignment. Therefore, the classification in Table 4 was conducted manually based on the results obtained by the algorithms; if outlier sequences are found within a given MSA, it is deemed *erroneous*; in case no outliers are detected, the alignment is considered as accurate and containing no errors. We evaluated the ability of each model to accurately identify erroneous MSAs within the test set; the study showed that our approach outperformed EvalMSA and OD-Seq by a significant margin: De-MISTED was able to detect 92% of all inaccurate MSAs with high accuracy compared to 81% by EvalMSA and 73% by OD-Seq. While the gap penalty based methods exhibited a similar performance in terms of precision (61%), our approach reached a higher precision value of **85%**.

6 Model Interpretability: A qualitative analysis

While classifying an MSA as containing at least one error constitutes the main objective of this work, it is important to assess whether our models' correct erroneous predictions are based on their ability to *correctly localize* the actual discrepancies within the alignment. Therefore, in order to interpret our classifiers' predictions, we produced saliency maps using a *post-hoc* visual explanation technique based on class activation mapping, called **Score-CAM** [52]. Score-CAMs provides insights on how the models achieved their classifications by highlighting the important regions in an input image that led them to their prediction. This technique also enables us to visually verify whether our model has succeeded in learning the underlying patterns in our dataset: if our models predict an MSA as containing at least one error, it should be able to identify the erroneous region within the image. To generate Score-CAMs, we input an image into the network, and chose the final convolutional layer to extract activation maps. The importance of each activation map is then measured by the Channel-wise Increase of Confidence (CIC). CIC upsamples each activation map and uses it as a mask on the input image to obtain its score on the target class. The final result is produced by a linear combination of score-based weights and activation maps. Let f be our model, and l denote the final convolutional layer. A_l denotes the activations of layer l, while A_l^k denotes the activation map for the k-th channel. For a given class of interest c, Score-CAM is defined as [52]:

$$L^c_{Score-CAM} = ReLU(\sum \alpha_k^c A_l^k) \tag{1}$$

where

$$\alpha_k^c = C(A_l^k) \tag{2}$$

and $C(.)$ denotes the CIC score for the activation map A_l^k.

TABLE 4
Performance comparison results. We evaluated our approach, De-MISTED, against EvalMSA [30] and OD-Seq [28] on the test set. De-MISTED yielded higher precision (85%) and recall (92%) in identifying erroneous MSAs.

	ERRORS		NO_ERRORS		
Method	Precision (%)	Recall (%)	Precision (%)	Recall (%)	Accuracy (%)
EvalMSA	61	81	67	42	63
OD-seq	61	73	62	50	62
De-MISTED (B)	**85**	**92**	**91**	**82**	**87**

In the following sections, we use Score-CAM to provide visual qualitative interpretations of our classification results on randomly selected MSA examples. We assess the ability of our models to correctly identify and localize inconsistencies in MSA images.

6.1 Identifying single errors in MSAs

Figure 7 shows two examples of Score-CAMs generated by models `(A)` (top) and `(B)` (bottom). The same alignments were used for both models and contain a single error: internal insertion in alignments (a) and (c) and a mismatch in alignments (b) and (d). All alignments were correctly predicted by the models as erroneous. As observed in the resulting Score-CAMs, both models succeeded in identifying the encountered error within each alignment; this is represented by the clear focus of the Score-CAM on the location of the detected errors. We observe that, in the obtained Score-CAMs by model `(A)`, the applied focus is spread over a large area surrounding the error. In Score-CAMs by model `(B)`, however, the focus outlines the exact location of the detected error. The following comment can be made with regard to this behavior: the *hybrid filters* enable model `(B)` to analyze the input image in vertical and horizontal contexts, which means it considers the relevant sequences as well as their vertical arrangement (*i.e* the sequences they are aligned with); this allows the model to extract more precise features about the location of the detected error within a given MSA. Further Score-CAM examples can be found in the supplementary material.

6.2 Identifying more than one error in MSAs

In order to assess the ability of our models to identify more than one error within a given MSA, we applied Score-CAM on MSAs containing *different types of errors*. The results are shown in Figure 8 for models `(A)` (top) and `(B)` (bottom). The same alignments were used for both models.

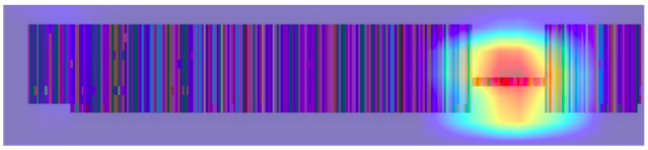
In Figure 8, models `(A)` and `(B)` correctly classify the MSAs as erroneous. In alignment (a), the obtained Score-CAM by model `(A)` highlights the existing deletion with a strong focus, while a weaker focus is visible on the location of the extension error. Alignment (b) contains three deletion errors, two of which were correctly identified by the model with an equally strong focus on the location of both errors. It is important to note that, as observed in the previous examples, all detected errors in the these examples were identified by highlighting a large area around the located errors. This behavior can also be observed in different examples presented throughout the remainder of this paper. In alignment (c), the deletion and extension errors are correctly identified by model `(B)`: the Score-CAM shows a strong focus on both errors. In alignment (d), we observe that model `(B)` succeeded in identifying all three deletion errors within the MSA (contrary to model `(A)`). Further Score-CAM examples can be found in the supplementary material.

It should be noted that, within certain MSAs, there exist gaps and constituent extended/inserted sequences that could very easily be misidentified as errors by our proposed models. Therefore, we applied Score-CAM on specific alignments to assess the ability of our classifiers to distinguish normal gaps/extensions/insertions from erroneous ones. The results are shown in Figure 9. Both models correctly classify the alignments as erroneous. In alignment (a), model `(A)` identifies the existing extension error on the far left by highlighting its location in the alignment. A second focus appears approximately in the middle of the alignment over a non-erroneous gap: this indicates that model `(A)` misidentified the normal gap as a deletion error. In alignment (b), model `(A)` identifies the existing deletion error (far right) with a strong focus; however, the resulting Score-CAM also shows a visible focus over certain sequence groupings of the MSA that are structurally similar to extension errors which the model misidentified as erroneous insertions. In contrast to model `(A)`'s behavior, the Score-CAM results produced by model `(B)` are comparatively more accurate: for both alignments, the model focuses solely on the existing (annotated) errors and disregards any similarities that could lead to a misidentification. Further examples can be found in the supplementary material.

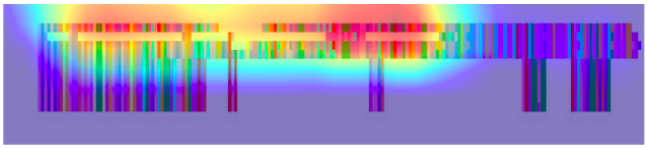
6.3 Identifying non-annotated errors

In the previous sections, we have demonstrated the ability of our classifiers to detect different types of annotated errors within MSAs. However, the statistical method SIBIS used to annotate the errors in the initial MSAs is not 100% accurate, and tends to have better specificity than sensitivity when detecting errors [8]. Consequently, some MSAs annotated as error free by SIBIS may actually contain errors which were overlooked by the annotation tool, as described in Section 5. Therefore, we used Score-CAM to investigate some of the 435 and 409 (from Figures 5 and 6) False Negative MSAs that were annotated as error-free, yet predicted by both models to contain at least one error. The results are shown in Figure 10.

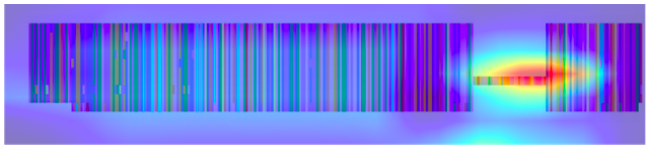
Upon close examination of the obtained results, the following comments are supported by human expertise: in example (a), while SIBIS deemed the MSA error-free, both models identified a deletion error on the far left whose existence was confirmed upon careful inspection. In alignment (b), model `(A)` (top) identified an extension and a deletion



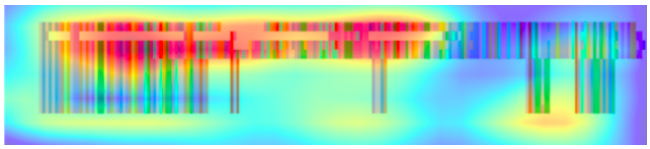
(a) MSA with an insertion error. Model (A) correctly classifies the alignment as erroneous and detects the location of the error.



(b) MSA with a mismatch error. Model (A) makes a correct prediction. The produced Score-CAM clearly identifies the location of the error.

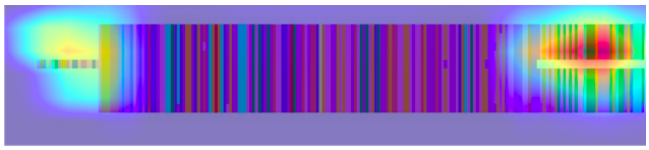


(c) MSA with an insertion error. Model (B) correctly classifies the alignment as erroneous and identifies the location of the error.

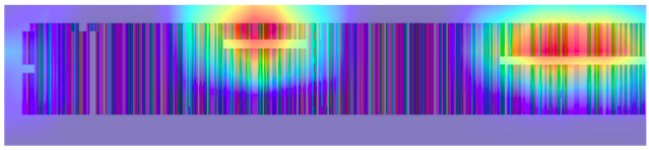


(d) MSA containing a mismatch error. Model (B) makes an accurate prediction. The produced Score-CAM correctly identifies the location of the error.

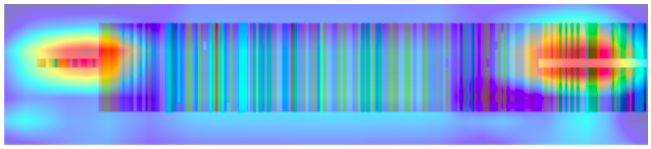
Fig. 7. Score-CAMs [52] produced using models (A) (top) and (B) (bottom) on single-error MSAs. Score-CAM highlights the portion of the image responsible for the model's final prediction.



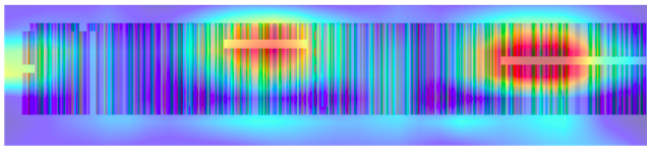
(a) MSA with two errors: an extension and a deletion. The MSA is correctly classified by model (A) as containing at least one error. The Score-CAM shows a strong focus on the deletion error, while a slightly weaker focus is applied on the extension.



(b) MSA with three deletion errors. Model (A) correctly identifies the alignment as erroneous. The Score-CAM highlights two deletions with a strong focus, and shows that model (A) fails to detect the third error.

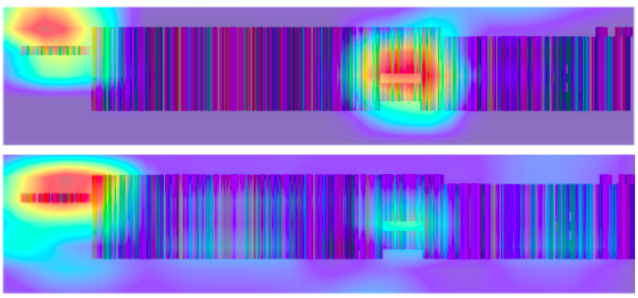


(c) MSA with two errors: an extension and a deletion. Model (B) correctly classifies the MSA as erroneous. Both errors are highlighted in the Score-CAM.

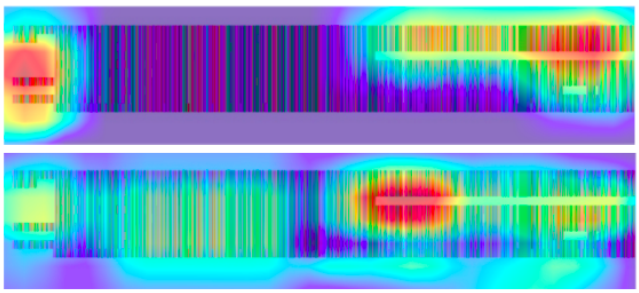


(d) MSA with three deletion errors. Model (B) correctly classifies the MSA and identifies the errors. Score-CAM strongly highlights two deletions and shows a weak focus on the third.

Fig. 8. Score-CAMs obtained by models (A) (top) and (B) (bottom) for MSAs containing more than one error. The models make accurate predictions and succeed in identifying most of the annotated errors.

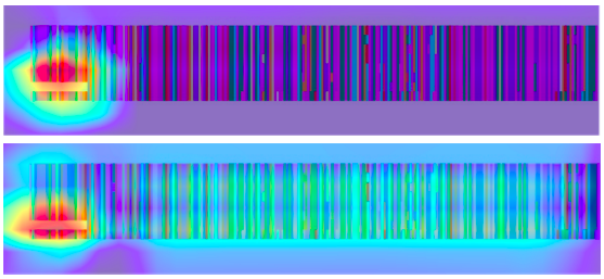


(a) MSA with a single extenstion error. Models (A) (top) and (B) (bottom) correctly classify the MSA as erroneous. Both models identify the existing extension error in the alignment. The Score-CAM obtained by model (A), however, highlights a non-erroneous gap misidentified as a deletion error.

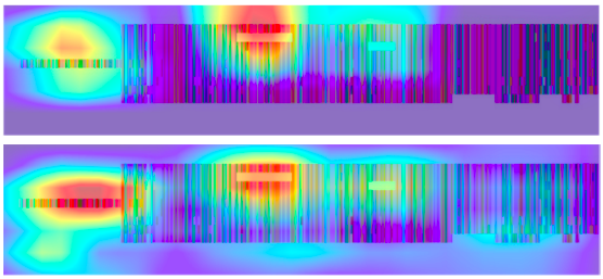


(b) MSA with a single deletion error. Both models correctly classify the alignment as erroneous. The existing deletion is identified by both models with a strong focus, as shown in the respective Score-CAMs. In the Score-CAM by model (A), however, a second focus is drawn to a non-erroneous extension.

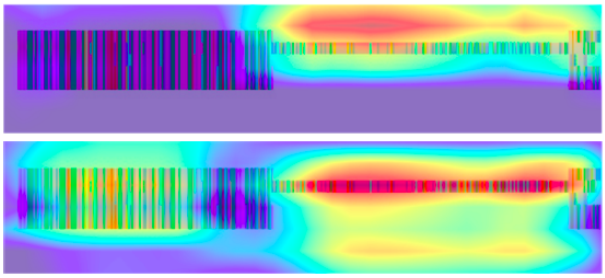
Fig. 9. A comparative analysis of Score-CAMs obtained by model (A) (top) and model (B) (bottom). We evaluate each model's ability to distinguish normal gaps/extensions/insertions from erroneous ones within a given MSA.



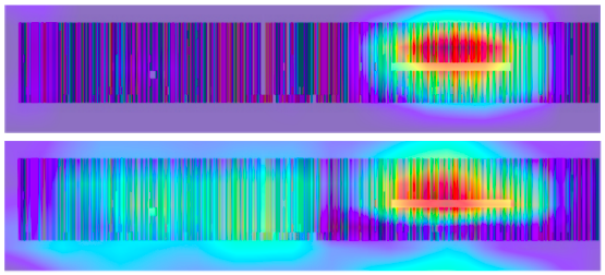
(a) MSA classified by models (A) (top) and (B) (bottom)as erroneous. The resulting Score-CAM by both models indicates the existence of a detection error.



(b) MSA classified as erroneous by both models. The obtained Score-CAMs highlight the existence of an extention error and two deletions.



(c) MSA predicted by both models as erroneous. The Score-CAMs highlight an identified insertion error.



(d) MSA classified by models (A) and (B) as containing at least one error. The obtained Sore-CAMs support this predcition by highlighting the existence of a deletion error.

Fig. 10. Initially error-free MSAs classified by models (A) (top) and (B) (bottom) as containing at least one error. The obtained Score-CAMs highlight the different errors identified by the models in the MSAs, which were missed by the original annotation tool.

error: while a strong focus of the obtained Score-CAM is observed over the deletion error, a weaker and slightly shifted focus indicates the existence of the extension error. The Score-CAM by model (B), however, strongly highlights the extension and deletion errors with a clear focus that outlines their respective locations. A third deletion error is identified by model (B) with a weak yet visible focus. While the models successfully identified the existence of said errors, a manually detected mismatch error was missed by both models as well as the annotation tool. The existence of all aforementioned errors is confirmed by human expertise. Example (c) shows that both models identified an insertion error that SIBIS has failed to detect. A thorough examination of the MSA confirms the existence of the detected insertion error. Lastly, a non-annotated deletion was detected by our models in alignment (d), whose existence in the alignment was manually confirmed. From these manual analyses, we conclude that the precision levels of 90% by model (A) and 91% by model (B) cited above are minimum values, and the true precision levels are probably higher than this.

7 Conclusion

Multiple Sequence Alignments are often riddled with different types of errors stemming either from inaccurate MSA-generating algorithms or from the constituent sequences. Since MSAs can be represented in the form of a colored output showing the similarity of aligned sequences, we hypothesized that this information could be exploited by Convolutional Neural Networks. To support our hypothesis, we proposed two CNN models, one of which implements a *hybrid filtering* within its convolutional layers, as an attempt to better consider both the horizontal and vertical context of an MSA.

Both models were tested on an in-house MSA dataset in which errors were carefully annotated. Our results show that both models exhibit good performance to distinguish erroneous MSAs from non-erroneous ones, obtaining 87% accuracy and 92% sensitivity. Although the metrics did not allow us to observe a significant difference between our two models, using a visualization technique on the activation maps of both models showed that the model implementing hybrid filtering more accurately identified the errors within MSAs. Overall, this qualitative analysis showed that both models can identify and localize the four kinds of errors found in MSAs: insertion, deletion, extension and mismatch. This further supports the idea that Convolutional Neural Networks are an appropriate approach to sort erroneous from non-erroneous MSAs.

These encouraging results open the possibility of designing multi-label classification models that can distinguish between the different kinds of errors found in MSAs and to accurately locate these errors within the sequences, in order to provide automated assistance to bioinformaticians working with Multiple Sequence Alignments. Moreover, as an attempt to widen the scope of portability of our models, we plan to use a transfer learning approach to transfer the knowledge learned by a model trained on the primate sequence dataset to different sequence sets.

Acknowledgments

The authors would like to thank the BiGEst bioinformatics platform for technical support. This work was supported by

French Infrastructure Institut Français de Bioinformatique (IFB) ANR-11-INBS-0013, ANR ArtIC ANR-20-THIA-0006 and Institute funds from the French Centre National de la Recherche Scientifique and the University of Strasbourg.

REFERENCES

[1] M. Chatzou, C. Magis, J.-M. Chang, C. Kemena, G. Bussotti, I. Erb, and C. Notredame, "Multiple sequence alignment modeling: methods and applications," *Briefings in bioinformatics*, vol. 2015, 11 2015.

[2] Y. Wang, H. Wu, and Y. Cai, "A benchmark study of sequence alignment methods for protein clustering," *BMC Bioinformatics*, vol. 19, 12 2018.

[3] J. D. Thompson, *Statistics for bioinformatics : methods for multiple sequence alignment*. iSTE Press, 2016.

[4] T. Warnow, *Revisiting Evaluation of Multiple Sequence Alignment Methods*, ser. Methods in Molecular Biology. Humana Press Inc., 2021, pp. 299–317.

[5] F. Prosdocimi, B. Linard, P. Pontarotti, O. Poch, and J. Thompson, "Controversies in modern evolutionary biology: the imperative for error detection and quality control," *BMC Genomics*, vol. 13, pp. 5 – 5, 2011.

[6] N. Scalzitti, A. Jeannin-Girardon, P. Collet, O. Poch, and J. Thompson, "A benchmark study of ab initio gene prediction methods in diverse eukaryotic organisms," *BMC Genomics*, vol. 21, p. 293, 04 2020.

[7] C. Meyer, N. Scalzitti, A. Jeannin-Girardon, P. Collet, O. Poch, and J. D. Thompson, "Understanding the causes of errors in eukaryotic protein-coding gene prediction: a case study of primate proteomes," *BMC Bioinformatics*, vol. 21, 2020.

[8] W. Khenoussi, R. Vanhoutreve, O. Poch, and J. Thompson, "Sibis: A bayesian model for inconsistent protein sequence estimation," *Bioinformatics (Oxford, England)*, vol. 30, 05 2014.

[9] R. Vanhoutreve, A. Kress, B. Legrand, H. Gass, O. Poch, and J. Thompson, "Leon-bis: Multiple alignment evaluation of sequence neighbours using a bayesian inference system," *BMC Bioinformatics*, vol. 17, 07 2016.

[10] M.-A. Dragan, I. Moghul, A. Priyam, C. Bustos, and Y. Wurm, "Genevalidator: Identify problems with protein-coding gene predictions," *Bioinformatics*, vol. 32, 01 2016.

[11] W. Pearson, "Finding protein and nucleotide similarities with fasta," *Current protocols in bioinformatics / editoral board, Andreas D. Baxevanis ... [et al.]*, vol. Chapter 3, 03 2004.

[12] D. Zaal and B. Nota, "Adoma: A command line tool to modify clustalw multiple alignment output," *Molecular Informatics*, vol. 35, 08 2015.

[13] R. C. Edgar and S. Batzoglou, "Multiple sequence alignment," *Current opinion in structural biology*, vol. 16 3, pp. 368–73, 2006.

[14] J. Thompson, J.-C. Thierry, and O. Poch, "Rascal: Rapid scanning and correction of multiple sequence alignments." *Bioinformatics (Oxford, England)*, vol. 19, pp. 1155–61, 07 2003.

[15] J. Thompson, F. Plewniak, and O. Poch, "Balibase: A benchmark alignment database for the evaluation of multiple alignment programs," *Bioinformatics (Oxford, England)*, vol. 15, pp. 87–8, 02 1999.

[16] J. Thompson, D. Higgins, and T. Gibson, "Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice." *Nucleic acids research*, vol. 22 22, pp. 4673–80, 1994.

[17] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, "Mafft: A novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic acids research*, vol. 30, pp. 3059–66, 08 2002.

[18] C. Notredame, D. G. Higgins, and J. Heringa, "T-coffee: A novel method for fast and accurate multiple sequence alignment." *Journal of molecular biology*, vol. 302 1, pp. 205–17, 2000.

[19] F. Corpet, F. Servant, J. Gouzy, and D. Kahn, "Prodom and prodom-cg: tools for protein domain analysis and whole genome comparisons," *Nucleic acids research*, vol. 28, pp. 267–9, 02 2000.

[20] J. Thompson, F. Plewniak, R. Ripp, J.-C. Thierry, and O. Poch, "Towards a reliable objective function for multiple sequence alignments," *Journal of Molecular Biology*, vol. 314, pp. 937–951, 12 2001.

[21] J. Tong, J. Pei, Z. Otwinowski, and N. Grishin, "Refinement by shifting secondary structure elements improves sequence alignments," *Proteins: Structure, Function, and Bioinformatics*, vol. 83, 12 2014.

[22] D. F. DeBlasio and J. Kececioglu, "Adaptive local realignment of protein sequences," *Journal of computational biology : a journal of computational molecular cell biology*, vol. 25 7, pp. 780–793, 2018.

[23] R. Gibbs, J. Rogers, M. Katze, R. Bumgarner, G. Weinstock, E. Mardis, K. Remington, R. Strausberg, J. Venter, R. Wilson, M. Batzer, C. Bustamante, E. Eichler, M. Hahn, R. Hardison, K. Makova, W. Miller, A. Milosavljevic, R. Palermo, and A. Zwieg, "Evolutionary and biomedical insights from the rhesus macaque genome," *Science*, vol. 316, pp. 222–34, 01 2007.

[24] A. Nagy and L. Patthy, "Mispred: A resource for identification of erroneous protein sequences in public databases," *Database : the journal of biological databases and curation*, vol. 2013, p. bat053, 01 2013.

[25] A. Nagy, H. Hegyi, K. Farkas, H. Tordai, E. Kozma, L. Banyai, and L. Patthy, "Identification and correction of abnormal, incomplete and mispredicted proteins in public databases," *BMC bioinformatics*, vol. 9, p. 353, 09 2008.

[26] J. D. Thompson, B. Linard, O. Lecompte, and O. Poch, "A comprehensive benchmark study of multiple sequence alignment methods: Current challenges and future perspectives," *PLoS ONE*, vol. 6, 2011.

[27] A. Nagy and L. Patthy, "Fixpred: a resource for correction of erroneous protein sequences," *Database: The Journal of Biological Databases and Curation*, vol. 2014, 2014.

[28] P. Jehl, F. Sievers, and D. Higgins, "Od-seq: Outlier detection in multiple sequence alignments," *BMC bioinformatics*, vol. 16, p. 269, 08 2015.

[29] R. D. Finn, A. Bateman, J. Clements, P. Coggill, R. Y. Eberhardt, S. R. Eddy, A. Heger, K. Hetherington, L. Holm, J. Mistry, E. L. L. Sonnhammer, J. Tate, and M. Punta, "Pfam: the protein families database," *Nucleic Acids Research*, vol. 42, no. D1, pp. D222–D230, 11 2014. [Online]. Available: https://doi.org/10.1093/nar/gkt1223

[30] A. Chiner-Oms and F. González-Candelas, "Evalmsa: A program to evaluate multiple sequence alignments and detect outliers," *Evolutionary Bioinformatics*, vol. 12, p. EBO.S40583, 2016, pMID: 27920488. [Online]. Available: https://doi.org/10.4137/EBO.S40583

[31] K. Tamura, G. Stecher, D. Peterson, A. Filipski, and S. Kumar, "Mega6: Molecular evolutionary genetics analysis version 6.0," *Molecular biology and evolution*, vol. 30, 10 2013.

[32] R. Jafari, M. Javidi, and M. Kuchaki Rafsanjani, "Using deep reinforcement learning approach for solving the multiple sequence alignment problem," *SN Applied Sciences*, vol. 1, 06 2019.

[33] M. Larkin, G. Blackshields, N. Brown, R. Chenna, P. McGettigan, H. Mcwilliam, F. Valentin, I. Wallace, A. Wilm, R. Lopez, J. Thompson, T. Gibson, and D. Higgins, "Clustal w and clustal x version 2.0," *Bioinformatics*, vol. 23, pp. 2947–2948, 01 2007.

[34] K. Katoh and D. Standley, "Katoh k, standley dm.. mafft multiple sequence alignment software version 7: Improvements in performance and usability. mol biol evol 30: 772-780," *Molecular biology and evolution*, vol. 30, 01 2013.

[35] X. Xuyu, Z. Dafan, J. Qin, and F. Yuanyuan, "Ant colony with genetic algorithm based on planar graph for multiple sequence alignment," *Information Technology Journal*, vol. 9, 02 2010.

[36] I.-G. Mircea, I. Bocicor, and G. Czibula, "A reinforcement learning based approach to multiple sequence alignment," in *Soft Computing Applications*, V. E. Balas, L. C. Jain, and M. M. Balas, Eds. Cham: Springer International Publishing, 2018, pp. 54–70.

[37] C. Kanz, P. Aldebert, N. Althorpe, W. Baker, A. Baldwin, K. Bates, P. Browne, A. Broek, M. Castro, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, F. Diez, N. Harte, T. Kulikova, Q. Lin, V. Lombard, and R. Apweiler, "The embl nucleotide sequence database," *Nucleic acids research*, vol. 33, pp. D29–33, 02 2005.

[38] H. Carroll, W. Beckstead, T. O'Connor, M. Ebbert, Q. Snell, and D. McClellan, "Dna reference alignment benchmarks based on tertiary structure of encoded proteins," *Bioinformatics (Oxford, England)*, vol. 23, pp. 2648–9, 11 2007.

[39] I.-G. Mircea, I. Bocicor, and G. Czibula, "A reinforcement learning based approach to multiple sequence alignment," in *Soft Computing Applications*, V. E. Balas, L. C. Jain, and M. M. Balas, Eds. Cham: Springer International Publishing, 2018, pp. 54–70.

[40] C. Zhang, W. Zheng, S. Mortuza, Y. Li, and Y. Zhang, "Deepmsa: Constructing deep multiple sequence alignment to improve contact prediction and fold-recognition for distant-homology proteins," *Bioinformatics (Oxford, England)*, vol. 36, 11 2019.

[41] G. Aoki and Y. Sakakibara, "Convolutional neural networks for classification of alignments of non-coding rna sequences," *Bioinformatics*, vol. 34, pp. i237 – i244, 2018.
[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84 – 90, 2012.
[43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.
[44] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.
[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.
[46] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
[47] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, pp. 211–252, 2015.
[48] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Y. Ding, A. Bagul, C. Langlotz, K. S. Shpanskaya, M. P. Lungren, and A. Y. Ng, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *CoRR*, vol. abs/1711.05225, 2017. [Online]. Available: http://arxiv.org/abs/1711.05225
[49] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3320–3328. [Online]. Available: http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf
[50] "wkhtmltopdf," https://wkhtmltopdf.org.
[51] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.
[52] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 111–119, 2020.

Hiba Khodji is a PhD student in Computer Science at the University of Strasbourg, France. She graduated from the National Institute of Statistics and Applied Economics in Rabat, Morocco, with a MSc in Information and Intelligent Systems. Her research interests include artificial intelligence, deep learning, and transfer learning.

Pierre Collet obtained his PhD in Computer Assisted Surgery at the University of Orsay in 1997, followed by post-doctoral internships on artificial evolution and genetic programming at INRIA and the Ecole Polytechnique. He was appointed Associate Professor at the Université du Littoral Côte d'Opale in 2003 and Professor at Strasbourg University in 2007. He is one of the founding members of the UNESCO UniTwin Digital Campus of Complex Systems. His current research interests include complex systems, stochastic optimization by artificial evolution and ant colony optimization applied to educational systems, massively parallel computing, and explicable and ethical artificial intelligence.

Julie Thompson obtained her PhD in Bioinformatics from Strasbourg University. After a number of years' experience in industry, she joined the European Molecular Biology Laboratory in Heidelberg as a computer engineer, before moving to the University of Strasbourg, where she is now a senior staff scientist. She has published more than 60 articles in scientific journals and according to the ISI Web of Science, she is one of the most highly cited researchers in her field with more than 120,000 citations. Her research is at the interface between computer science and computational biology, and she specializes in 'big data' integration and management, artificial intelligence approaches for bioinformatics, genome sequence analysis and evolution, and inference of genotype-phenotype relations in the study of rare genetic diseases.

Anne Jeannin-Girardon obtained her PhD in Computer Science at the Université de Bretagne Occidentale in 2014. She then did a postdoctoral internship at Stony Brook University, USA on modeling of the immune system, before joining the University of Strasbourg as Associate Professor in 2016. Her current research focuses on the study of Complex Systems and Artificial Intelligence, particularly Deep Learning and the notions of autonomy, explainability and ethics.