iMotor-CNN: Identifying molecular functions of cytoskeleton motor proteins using 2D convolutional neural network via Chou’s 5-step rule

Nguyen Quoc Khanh Le*, Edward Kien Yee Yappb, Yu-Yen Ou, Hui-Yuan Yeh

ARTICLE INFO

Keywords:
Cytoskeletal filaments
Protein function prediction
Dynein and kinesin
Myosin light chain
Deep learning
Position specific scoring matrix

ABSTRACT

Motor proteins are the driving force behind muscle contraction and are responsible for the active transportation of most proteins and vesicles in the cytoplasm. There are three superfamilies of cytoskeletal motor proteins with various molecular functions and structures: dynein, kinesin, and myosin. The functional loss of a specific motor protein molecular function has linked to a variety of human diseases, e.g., Charcot-Marie-Tooth disease, kidney disease. Therefore, creating a precise model to classify motor proteins is essential for helping biologists understand their molecular functions and design drug targets according to their impact on human diseases. Here we attempt to classify cytoskeleton motor proteins using deep learning, which has been increasingly and widely used to address numerous problems in a variety of fields resulting in state-of-the-art results. Our effective deep convolutional neural network is able to achieve an independent test accuracy of 97.5%, 96.4%, and 96.1% for each superfamily, respectively. Compared to other state-of-the-art methods, our approach showed a significant improvement in performance across a range of evaluation metrics. Through the proposed study, we provide an effective model for classifying motor proteins and a basis for further research that can enhance the performance of protein function classification using deep learning.

1. Introduction

Motor proteins are the driving force behind muscle contraction, and they play an active and essential role in transporting proteins and vesicles in the cytoplasm. These proteins can convert the chemical energy of adenosine triphosphate (ATP) hydrolysis into mechanical work which moves along actin filaments or microtubules. There are three superfamilies of cytoskeletal motor proteins: dynein, kinesin, and myosin (Fig. 1). The dynein and kinesin superfamilies are microtubule motors that move vesicles and organelles within cells, leading to the whipping of flagella and cilia, and act within the mitotic and meiotic spindles to discriminate duplicated chromosomes [1–3]. Various biological processes characterize the force and movement that dynein proteins enact on microtubules, including ciliary beating, cell division, and intracellular transport [4]. Kinesin superfamily proteins play a vital role in transporting various cargoes directly, including membranous organelles, protein complexes, and mRNAs [5]. Myosin motors act upon actin filaments to produce cell surface contractions and alternative morphological differences, including vesicle motility, cytoplasmic streaming and muscle cell contraction [6]. Numerous types of motor proteins have now been identified in humans, and several studies have demonstrated that a functional loss of these proteins has implication of the formation of diseases [7]. For instance, kinesin and cytoplasmic dynein are found in spinal spheroids which have links to motor neuron disease [8]. Mutations in kinesin and targeted disruption of kinesin functions have also been linked to neurodegenerative diseases [9,10]. Myosin-induced acute myocarditis is a T-cell-mediated disease [11]. Myosin IXB variant increases the risk of celiac disease and points toward a primary intestinal barrier defect [12].

Motor proteins play an important role in human function which has piqued the interest of numerous bioinformatics researchers. For instance, Miki et al. [13] constructed an analysis of the kinesin superfamily of motor proteins, in which they focus their insights on their structure and function. Next, Yagi [14] presented the bioinformatics method for classifying dynein heavy chains. Khataee and Liew [15] proposed a mathematical model to study the mechanical kinetics of forward and backward stepping of kinesin motor based on the four-state discrete stochastic model of the motor. They also presented a stochastic...
Automaton model for simulating kinesin processivity [16]. Stedman

tides (shown in pink) between each molecular function are di-
catalytic domains are shown in orange, whereas the stalks, which form ex-
Each motor protein's molecular function is a dimer of two heavy chains. Motor

Fig. 1. Overview of three superfamilies of cytoskeleton motor proteins. (For interpretation of the re-
later chain in myosin, two light chains in kinesin, and a complex set of intermediate, light-intermediate and light chains in dynein). (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

automaton model for simulating kinesin processivity [16]. Stedman et al. [17] used bioinformatics techniques to prove that the myosin gene mutation correlated with anatomical changes in the human lineage. There were also several previous bioinformatics studies to investigate the myosin motor domain or myosin phosphatase subunit diversity [18,19]. No study, to our knowledge, has considered classifying the motor proteins into their superfamilies (such as dynein, kinesin, and myosin). A new approach is therefore needed to classify them, and our research thus aims to find a solution to this problem.

In earlier years, numerous researchers attempted to use shallow neural networks to resolve various problems in bioinformatics, especially in the classification of protein functions. For example, Weka is a machine learning tool for the general use of data mining techniques, and it was also used in many bioinformatics studies [20]. Next, several problems on protein function classification achieved high performance results by using radial basis function (RBF) networks [21,22]. Moreover, the LibSVM package [23] is a library for Support Vector Machines, which presents itself as a guide for biologists to implement learning algorithms in their field of work. Most of the time, using traditional machine learning techniques have achieved valuable results. However, the emergence of deep learning resulted in the need to increase the performance in bioinformatics fields. Deep learning is an advanced form of machine learning which makes use of artificial intelligence technique to learn representative data with multiple layers of neural networks [24]. There are a lot of advantages when applying deep learning such as achieving state-of-the-art results, the reduction of the need for feature extraction, and an increase in the speed of computation time using a graphics processing unit (GPU). Recently, there is a transition from traditional machine learning to deep learning techniques in bioinformatics in general and protein functions in particular. For instance, Alipanahi et al. [25] attempted to predict the sequence specificities of DNA and RNA binding proteins by deep learning. Spencer et al. [26] proposed a deep learning network for ab initio protein secondary structure prediction. Almagro Armenteros et al. [27] presented DeepLoc, a deep learning framework for predicting protein subcellular localization.

Based on the advantages of deep learning, this study consequently proposes the use of a 2D convolutional neural network (CNN) constructed from position specific scoring matrix (PSSM) profiles to classify cytoskeleton motor proteins. The main achievements, including contributions to the field are presented as follows: (i) development of a deep learning framework to classify molecular functions from protein sequences, to which our model exhibited a significant improvement beyond traditional machine learning algorithms; (ii) first computational study to classify motor proteins into their molecular functions and provide useful information to biologists to discover the motor superfamilies; (iii) valid benchmark dataset to train and test motor proteins with high accuracy; and (iv) introduction of newly discovered cytoskeleton motor proteins in Homo sapiens and Arabidopsis thaliana, which forms a basis for future research on motor proteins.

As shown in a series of recent publications [28–36], to develop a really useful sequence-based statistical predictor for a biological or biomedical system, one should observe the guidelines of 5-step rule [37]. We would restate the five steps here for clarity: (i) how to construct or select a valid benchmark dataset to train and test the predictor; (ii) how to formulate the biological sequence samples with an effective mathematical expression that can truly reflect their intrinsic correlation with the target to be predicted; (iii) how to introduce or develop a powerful algorithm (or engine) to operate the prediction; (iv) how to properly perform cross-validation tests to objectively evaluate the anticipated accuracy of the predictor; (v) how to establish a user-friendly web-server for the predictor that is accessible to the public. Below, we would describe how the process of dealing with these steps one at a time.

2. Materials and methods

We propose a novel approach using 2D CNN and PSSM profiles to classify the molecular functions of cytoskeleton motor proteins with high performance. Fig. 2 illustrates a flowchart of the study, which is made up of four subprocesses: data collection, feature set extraction, generation, and model evaluation. The detailed description of the proposed approach would be discussed in the following paragraphs.

2.1. Data collection

Two extensive resources for gene and protein sequences UniProt (release 2017_12) [38] and the latest version (2017) of Gene Ontology [39,40] were used to collect the data. In Gene Ontology, three pieces of genomic information that can be collected are cellular component, molecular function, and biological process. In this study, we used the molecular function annotation information to obtain the motor proteins dataset. The collected molecular functions for three classes of motor proteins are dynein, kinesin, and myosin. In this section, we chose all of the proteins with the experimental evidence used in a manual assertion. Next, we removed redundant data between the three types of datasets. Thereafter, BLAST [41] was applied to eliminate redundant sequences with a similarity of more than 30%. Consequently, 721 dynein proteins, 469 kinesin proteins, and 490 myosin proteins remained and they were used to construct the model. The objective of this step is to avoid the problem of overfitting in our model.

To solve the multi-classification problem, we divided it into three binary problems, in which we used one class as the positive data and the rest as the negative data. We then repeated the experiment two more times, with each of the other two classes as the positive data and the rest as the negative data. We divided all the collected sequences into two datasets: cross-validation and independent dataset. The cross-validation dataset was used for constructing our model, and the independent dataset was used for evaluating the performance of the proposed method. Table 1 lists all the details of the dataset used in this study. Our source code and dataset are also freely available for download at: https://github.com/khanhilee/deep-motor.
2.2. Feature extraction for identifying three superfamilies of cytoskeleton proteins

With the explosive growth of biological sequences in the post-genomic era, one of the most important but also most difficult problem in computational biology is how to express a biological sequence with a discrete model or a vector, yet still being able to keep considerable sequence-order information or key pattern characteristic. This is because all the existing machine-learning algorithms (such as “Optimization” algorithm [42], “Covariance Discriminant” or “CD” algorithm [43,44], “Nearest Neighbor” or “NN” algorithm [45], and “Support Vector Machine” or “SVM” algorithm [45,46]) can only handle vectors, as elaborated in a comprehensive review [47]. However, a vector defined in a discrete model may completely lose all the sequence-pattern information. In order to avoid completely losing the sequence-pattern information for proteins, the pseudo amino acid composition (PseAAC) [48] was proposed. Ever since the concept of Chou’s PseAAC was proposed, it has been widely used in nearly all the areas of computational proteomics (e.g., a long list of references cited in Ref. [49]). Because it has been widely and increasingly used, recently three powerful open access soft-wares, called ‘PseAAC-Builder’ [50], ‘propy’ [51], and ‘PseAAC-General’ [52], were established: the former two are for generating various modes of Chou’s special PseAAC; while
the 3rd one for those of Chou's general PseAAC [37], including not only all the special modes of feature vectors for proteins but also the higher level feature vectors such as “Functional Domain” mode (see Eqs. (9) and (10) of [37]), “Gene Ontology” mode (see Eqs.11-12 of [37]), and “Sequential Evolution” or “PSSM” mode (see Eqs.13-14 of [37]). Encouraged by the successes of using PseAAC to deal with protein/peptide sequences, the concept of PseKNC (Pseudo K-tuple Nucleotide Composition) [53] was developed for generating various feature vectors for DNA/RNA sequences that have proved very useful as well. Particularly, Pse-in-One2.0 [55] was established to be able to generate any desired pseudo components feature vectors for protein/peptide and DNA/RNA sequences according to the need of users' studies.

The feature extraction used in this study is PSSM, which is a matrix designed for all amino acid motifs in protein sequences. The PSSM profile has been successfully used in numerous bioinformatics applications; for example in membrane types [56], Rab GTPases [57], and binding sites [58] predictions with significant improvements. We created the PSSM profiles from all FASTA sequences by using PSI-BLAST [41] and the non-redundant (NR) protein database. In order to recognize the superfamilies of motor proteins, we had to determine the most dependable protein sequence for each amino acid. We located 20 types of amino acids in the corrected sequences, leading to the occurrence of a matrix. Each value of the 400D input vector was divided by the sequence length and then normalized by using the formula:

$$f(x) = \frac{1}{1 + e^{-x}}$$

(1)

2.3. Two-dimensional convolutional neural network architecture

The CNN is made up of three different layers: input layers, hidden layers, and output layers. The details of each layer would be explained below.

2.3.1. Input layers

We conducted this study using a 2D convolutional neural network, which is a conventional neural network for deep learning. 2D CNN has been applied in numerous studies in various fields with remarkable results, i.e., image classification [59], face recognition [60], and natural language processing [61]. The bottom part of Fig. 2 illustrates the layer structure of the simplified convolutional neural network. Our deep learning architecture was carried out using the Keras library with Theano backend [62]. GPU computing and CUDA kernel were also applied to accelerate the performance more efficiently. In this study, the input layer parameters were from PSSM profiles which were converted into 20 × 20 matrices. By using these matrices as the input data, we aim to propose a method to classify motor proteins into distinct molecular functions. We assumed the 20 × 20 matrix to be an image with 20 × 20 pixels so that we could train the 2D CNN model with different weights and biases to enhance its predictive performance. The purpose of using a 2D CNN model is to capture the hidden features inside the PSSM profiles as opposed to a 1D structure.

2.3.2. Hidden layers

With more hidden layers generated, similarly more hidden features would also be generated in the CNN, thereby allowing easier classification of motor proteins. We constructed the hidden layers using various sub-layers: zero padding 2D layers, convolutional 2D layers, max pooling 2D layers and fully-connected layers with various parameters. All of the layers were being combined in a specific order to become the nodes in the CNN architecture. In the first few layers of deep neural networks, we would like to preserve as many hidden patterns about the original data, so we apply the zero padding layers. This layer can add zero values from the input. In our model the stride is s = 1, so the size of zero padding (zp) is given by the formula:

$$zp = \frac{k - 1}{2}$$

(2)

where k (= 3) is the filter size.

Subsequently, we applied convolution to the 2D matrices with a 3 × 3 sliding window, the features were learned with the small 3 × 3 matrices and shifted one unit at a time. Each neuron received inputs with the weights and biases from the previous layer, and they were trained again. The bottom part of Fig. 2 indicates the process of 2D convolutional layers. Convolutional layers are the core building block of a CNN to perform most of the computational heavy lifting. To compute the output size for a convolutional layer, we applied the formula:

$$o_s = \frac{w - k + 2p}{s} + 1$$

(3)

where w is the input size. Note that the padding is p = 0 as the padding and convolution are performed in two separate steps.

There are various types of activations in the convolutional layers such as linear, sigmoid, tanh, and so on, but in this study, we used rectified linear unit (ReLU). ReLU is the most important activation function for all deep neural networks and became popular in the last few years. The ReLU activation function is defined by the formula:

$$f(x) = (0, x)$$

(4)

where x is the number of inputs into a neural network.

The max pooling layers were frequently embedded inside the convolutional layers to decrease the dimensional size of performing arithmetic in the network and limit overfitting. In this study, we used a generally known design of two pooling strides with 2 × 2 filters.

2.3.3. Output layer

The first few layers in the output layer are flatten layers. Because the output layers require the distribution of all classes as probabilities, the flatten layers convert the input matrix into a vector. It is believed that this output can then be used in the following layers to generate information. Subsequently, we see a dense layer, which is a fully connected neural network. In this layer, the classification will be accomplished on the features from the convolutional layers and the pooling layers. Including a fully connected layer is a typical approach of learning non-linear hybrids of the features. Moreover, the dropout layer is a regularization technique proposed to improve the classifying performance of our model and avoid overfitting issues [63]. In the dropout layer, the model will randomly deactivate the neurons in a layer with a certain probability p. If the dropout value is added in a layer, the neural network will ignore selected neurons during training, thus ensuring the training time to be faster. In this study, the dropout values ranging from 0 to 1 are used to evaluate our model. Furthermore, the ReLU again plays an important role as an activation function used during the construction of the CNN to classify motor superfamilies.

The last layer of our CNN architecture is a softmax (normalized exponential function), which is a fully connected layer containing a logistic function defined by the formula:

$$\sigma(z)_i = \frac{e^{zi}}{\sum_k e^{zk}}$$

(5)

where z is a K-dimensional input vector, and \(\sigma(z)\) is a K-dimensional vector of real values in the range of [0,1] where the ith element is the predicted probability of the ith class from the sample vector x. The softmax layer is used in numerous multiclass classification methods and neural networks to highlight significant values from the input. In summary, we set a total of 44642 trainable parameters in the model matrices, the output volume would be 22 × 22. The zero padding layer allowed our model to not have a different output dimensions after applying filters to the input data. In our model the stride is s = 1, so the size of zero padding (zp) is given by the formula:
Table 2
All layers and trainable parameters of the two-dimensional convolutional neural networks in this study.

<table>
<thead>
<tr>
<th>Layer (type)</th>
<th>Output shape</th>
<th>Param #</th>
<th>Connected to</th>
</tr>
</thead>
<tbody>
<tr>
<td>zeropadding2d_1</td>
<td>(22, 22, 1)</td>
<td>0</td>
<td>zeropadding2d_input_1</td>
</tr>
<tr>
<td>convolution2d_1</td>
<td>(20, 20, 32)</td>
<td>320</td>
<td>convolution2d_1</td>
</tr>
<tr>
<td>maxpooling2d_1</td>
<td>(10, 10, 32)</td>
<td>0</td>
<td>maxpooling2d_1</td>
</tr>
<tr>
<td>zeropadding2d_2</td>
<td>(12, 12, 32)</td>
<td>0</td>
<td>maxpooling2d_2</td>
</tr>
<tr>
<td>convolution2d_2</td>
<td>(10, 10, 32)</td>
<td>9248</td>
<td>convolution2d_2</td>
</tr>
<tr>
<td>maxpooling2d_2</td>
<td>(5, 5, 32)</td>
<td>0</td>
<td>maxpooling2d_2</td>
</tr>
<tr>
<td>zeropadding2d_3</td>
<td>(7, 7, 32)</td>
<td>0</td>
<td>maxpooling2d_2</td>
</tr>
<tr>
<td>convolution2d_3</td>
<td>(5, 5, 64)</td>
<td>18496</td>
<td>convolution2d_3</td>
</tr>
<tr>
<td>maxpooling2d_3</td>
<td>(2, 2, 64)</td>
<td>0</td>
<td>convolution2d_3</td>
</tr>
<tr>
<td>flatten_1</td>
<td>(1, 256)</td>
<td>0</td>
<td>flatten_1</td>
</tr>
<tr>
<td>dropout_1</td>
<td>(1, 256)</td>
<td>0</td>
<td>dropout_1</td>
</tr>
<tr>
<td>dense_1</td>
<td>(1, 64)</td>
<td>16448</td>
<td>dense_1</td>
</tr>
<tr>
<td>dense_2</td>
<td>(1, 2)</td>
<td>130</td>
<td>dense_1</td>
</tr>
<tr>
<td>activation_1</td>
<td>(1, 2)</td>
<td>0</td>
<td>activation_1</td>
</tr>
</tbody>
</table>

(38)

2.4. Performance evaluation

Although the jackknife test is an approximately unbiased estimator of the generalization performance, it has two major drawbacks, e.g. a high variance (since all the datasets used for the estimate are very similar to each other), and it is expensive to calculate (it requires \( n \) estimates, where \( n \) is the number of observations in the dataset). It is therefore also proven that the 5 or 10 fold cross-validation is a good compromise between impartiality and computational requirements. Not only that, other forms of resources to learn more about the cross-validation process are being highlighted [64]. Likewise, we first trained the model by applying 5-fold cross-validation technique to the whole training dataset. Due to the 5-fold cross-validation yielding different results every time, we implemented ten times of 5-fold cross-validation to generate more reliable results. The cross-validation result was being calculated through the averaging of all the ten 5-fold cross-validation results. Based on the 5-fold cross-validation results, hyper-parameter optimization process was employed to find the best model for each dataset. Furthermore, to prevent any systematic bias in the cross-validation set, the independent dataset was re-used to evaluate the performance accuracy.

Although the traditional metrics copied from math books were often used in literature to measure the prediction quality of a prediction method, they are no longer good because they lack intuitiveness and they are easily comprehensible equations for most biologists. Particularly the Matthews correlation coefficient (MCC), which is a very important metric used for reflecting the stability of a prediction method. Fortunately, based on the Chou’s symbols introduced for studying protein signal peptides [65], a set of four intuitive metrics were derived, as given in Eqs. (6)–(9). Ever since then, the new set of intuitive metrics have been concurred and admired by a series of recent publications (see, e.g. Refs. [34,35,66,67]). However, it is instructive to point out that the metrics (as defined in Eqs. (6)–(9) and their original forms) are valid only for single label systems; for the multi-label systems (where a sample may simultaneously belong to several different classes), whose existence has become more frequent in system biology, system medicine and biomedicine, a completely different set of metrics as defined in Ref. [68] is absolutely needed. In a bid to evaluate the performance of the methods employed, we also adopted this new set of intuitive metrics. Some standard metrics were used, such as sensitivity, specificity, accuracy, and MCC using below given formulae (TP, FP, FN are true positive, false positive, true negative, and false negative values, respectively):

\[
\text{Sensitivity} = 1 - \frac{N^-}{N^*}, \quad 0 \leq \text{Sen} \leq 1
\]

\[
\text{Specificity} = 1 - \frac{N^+}{N^-}, \quad 0 \leq \text{Spec} \leq 1
\]

\[
\text{Accuracy} = 1 - \frac{N^+ + N^-}{N^* + N^-}, \quad 0 \leq \text{Acc} \leq 1
\]

\[
\text{MCC} = \frac{1 - \frac{N^+ + N^-}{N^* + N^-}}{\sqrt{(1 + \frac{N^- - N^*}{N^-})(1 + \frac{N^* - N^-}{N^-})}}, \quad -1 \leq \text{MCC} \leq 1
\]

The relations between these symbols and the symbols in Eqs. (6)–(9) are given by:

\[
\begin{align*}
N^+ &= TP \\
N^- &= FN \\
N^* &= TP + N^+ \\
N^- &= TN + N^-
\end{align*}
\]

We also used the Receiver Operating Characteristic (ROC) curves to further illustrate our model in various experiments. In addition, the area under the ROC curve (AUC) metric is a scalar value that represents the overall performance of the model [69]. The AUC score is always bounded between zero and one, and there is no realistic classification with an AUC less than 0.5. The AUC metric is therefore used to compare the efficiency of different models.

3. Results and discussions

3.1. Performance of classifying cytoskeleton motor proteins with traditional machine learning techniques

We have classified the molecular functions of motor proteins using various traditional machine learning algorithms, i.e., kNN [70], RandomForest [71], LibSVM [23], and QuickRBF [72]. All the processes for tuning parameters had been carried out on the training dataset and the optimizations had been chosen according to the accuracy metric. We varied the number of nearest neighbors in kNN from one to ten, performed a grid search to estimate the accuracy of each parameter combination to find the optimal cost and gamma in LibSVM (log2c was ranged from \(-5\) to \(5\), log2g was ranged from \(-4\) to \(0\)), number of trees were ranged from 100 to 500 in RandomForest; and the bandwidth set to five in QuickRBF (this bandwidth value has been used in a variety of bioinformatics applications with positive results [58,72,73]).

Table 3 shows the experimental results from PSSM profile features with the 5-fold cross validation and an independent dataset. We observe that the LibSVM classifier achieved the highest results in dynemin, kinesin, and myosin with an accuracy of 92.1%, 90.9%, and 89.4%, respectively. For the independent dataset, the accuracy came to 97.5%, 96.1%, and 96.1%, respectively. Therefore, the LibSVM could be applied for classifying these three classes of cytoskeleton motor proteins.

3.2. Performance of classifying the molecular functions of motor proteins with 2D convolutional neural networks

The results from LibSVM classifier in the previous section were high, but we would like to improve the performance results using 2D CNN. First, we investigated the sensitivity of the results to the number of filters—32, 64, 128 and 256—and we found 64 filters to be the best option (as shown in Supplementary Table S1). Therefore, to develop the precise model, we applied 64 filters in our hidden layer. Obviously, the number of epochs has a direct impact on the convergence of the neural network, a phenomenon proven in a variety of studies [74–76]. Thereafter, we ran the experiments for a suitably large number of epochs (200) in this study. During the running process, we saved the model after every epoch and then kept the epoch number that improved the predictive performance. We then optimized the neural networks using a variety of optimizers: rmsprop, adam, nadam, sgd, and
adadelta. The model was reinitialized, i.e. a new network is built, after each round of optimization so as to provide a fair comparison between the different optimizers. Overall, adadelta and rmsprop performed better than the others as shown in Fig. 3. A possible explanation is that these two optimizers compute an adaptive learning rate for each parameter, while the other optimizers use a constant learning rate resulting in a lower accuracy and instability. Note that for SGD the learning rate has decreased to 0.001 due to the chaotic and unstable results for its default value of 0.01. We chose to use adadelta, one of two higher optimizers in our final model structure. Adadelta was also used in previous work [77] with the same data type of PSSM profiles.

The last parameter applied in this section is the dropout [63], which was intended to enhance the performance of the neural network and to avoid overfitting inside the dataset. As shown in Supplementary Table S2, the best dropout value for dynein, kinesin and myosin is 0.5, 0.2 and 0.3, respectively. In summary, we applied 64 filter layers, 200 epochs, and optimal dropout values to classify the motor superfamilies.

3.3. Comparative performance between 2D CNN and the other neural networks

In this section, we aim to compare our 2D CNN performance results with the other neural networks. All of the comparative performance results are shown in Table 4. The first one is used to compare the traditional machine learning results (refers to the best performing traditional machine learning technique studied: SVM) with the deep learning results. We observe that our 2D CNN exhibited a higher performance than the other traditional machine learning techniques across most evaluation metrics. It thus can be claimed that the deeper networks (with more hidden layers) would generate more hidden feature sets than the shallow networks [23,70–72]. Moreover, CNNs assume that the input to the network contains significant information in the arrangement of data in a matrix. In contrast, the arrangement of the rows and columns in the PSSM profiles does not seem to be important after preprocessing. Therefore, we investigated similar neural network-based approaches that are not sensitive to artificial spatial information.

Table 3

<table>
<thead>
<tr>
<th></th>
<th>Dynein</th>
<th>Kinesin</th>
<th>Myosin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sen</td>
<td>Spe</td>
<td>Acc</td>
</tr>
<tr>
<td>Cross-validation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kNN</td>
<td>85</td>
<td>89.2</td>
<td>87.4</td>
</tr>
<tr>
<td>RF</td>
<td>82.1</td>
<td>93.9</td>
<td>88.8</td>
</tr>
<tr>
<td>SVM</td>
<td>87.9</td>
<td>95.3</td>
<td>92.1</td>
</tr>
<tr>
<td>RBF</td>
<td>87.5</td>
<td>95.3</td>
<td>91.9</td>
</tr>
<tr>
<td>Independent</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kNN</td>
<td>90.8</td>
<td>94.3</td>
<td>92.8</td>
</tr>
<tr>
<td>RF</td>
<td>89.2</td>
<td>95</td>
<td>92.5</td>
</tr>
<tr>
<td>SVM</td>
<td>97.5</td>
<td>97.5</td>
<td>97.5</td>
</tr>
<tr>
<td>RBF</td>
<td>95.8</td>
<td>96.9</td>
<td>96.4</td>
</tr>
</tbody>
</table>

(Sen: Sensitivity, Spe: Specificity, Acc: Accuracy, MCC: Matthew’s correlation coefficient).

Fig. 3. The validation accuracy on classifying motor proteins using different optimizers. The performance from adadelta optimizer (green line) was generally higher and more consistent than other optimizers. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)
in the PSSMs to confirm that the network focuses on the raw data and not artifacts in the matrix structure. There are two deep neural network architectures chosen, e.g., 1D CNN and fully connected neural network (FCNN). As shown in Table 4, our 2D CNN results outperformed the other two deep neural networks at the same network level. Therefore, in this problem that we are facing, applying 2D CNN on PSSM matrices would be able to capture more features analogous to images, a function that the other networks cannot. Consequently, there was an increase in the performance results which enabled the efficient classification of the motor superfamilies.

Here, a much-unexpected result is that the performance on the independent set is significantly higher than the performance during the cross-validation of this study. This means we can learn a lot in the training set and apply them to unseen data. As expressed in its fundamental definition, deep learning method is about performance on unseen data, thus there unexpected results are bound to happen as there are no guarantees. When addressing this issue, we have to note that we have an excess of data to train the model in the independent dataset, thereby increasing the performance in the independent dataset. Another reason is that we had 400 features but only about 1400 samples so the increase in the number of samples is a solution that would create a more consistent model for the future.

### 3.4. Important features inside PSSM profiles

Because we transformed the original PSSM profile to 400D PSSM profiles, we no longer retain the original protein sequence as well as the domain information in the protein for inclusion in the CNN model. In this section, we used a feature selection technique called F-score [78] for the purpose of identifying features that have the greatest contribution towards improving the outcome of the problem. This technique was successfully used in many similar problems [21,22]. The idea is to find out differences between the dynein, kinesin and myosin PSSM matrices, to which our model would then capitalize on them to generate better results. As shown in Supplementary Fig. S1, we plotted the feature maps of these classifications and easily observed that there were differences between the three proteins. For dynein classification, the feature with the greatest contribution is in the amino acid F column, especially at the motif FF. On the other hand, the motifs with the greatest contribution are CR and GM for kinesin and myosin classification, respectively. All of these motifs might play essential roles in deciding the functions of cytoskeleton motor proteins and our model identified them as important hidden features. In summary, we found that there were differences among the three proteins and based on this we attempted to divide the multi-classification problem into three binary classifications. It would be of aid to us to acquire the most important features according to the specific protein and achieve the best results for each one.

### 3.5. Multi-classification results for classifying cytoskeleton motor proteins

We also compared the performance between the binary and multi-classification problem in this study. To investigate the performance of 2D CNN via multiple classification, ROC curve and AUC were used in this section. Fig. 4 illustrates the ROC curve for multi-classification of cytoskeleton motor proteins with 2D CNN. We easily observe that the ROC curve in all three classes almost reached the significant point (AUC = 0.99, 0.96, and 0.98, respectively). Compared to the binary classification (Supplementary Fig. S2), the performance results were also at the same high level.

We also report the values of the metrics in Eqs. (6)–(9) for the corresponding multi-classification problem as shown in Table 5. The results are only slightly lower as compared to binary classification. This suggests that our deep neural network architecture could achieve a high performance even using multi-classification; however, more data is required to further test this finding.

### 3.6. Classifying new cytoskeleton motor proteins in genomic sequences using proposed method

We applied our method to explore the cytoskeleton motor proteins in newly discovered genomes from UniProt [38] which contains all of the reviewed sequences. This finding is necessary for identifying new motor molecular functions from unknown genomes. We chose two genomes, *Homo sapiens* (human), and *Arabidopsis thaliana*, which are two common organisms that play an essential role in biological studies. Because we intend to discover only new proteins, we chose cytoskeleton motor proteins annotated after 2017. After using our model (by using multiple binary classification CNNs) to classify this dataset, we found novel proteins with motor molecular function annotations. Because our model accomplished an accuracy of more than 90%, these proteins can be elucidated as motor molecular functions. To improve on the accuracy of our prediction, we used our model to predict cytoskeleton motor proteins and benchmark them against the automatic systems annotating UniProt/TrEMBL. Thereafter, we gathered the statistics on how similar and dissimilar our model performed against UniProt pipelines as well as how many newly predicted proteins our model can predict as shown in Table 6. Compared with the UniProt pipeline, our model has a similarity of 68.4% (13 out of 19 proteins) and 85.7% (78 out of 91 proteins) for *Homo sapiens* and *Arabidopsis thaliana*, respectively. We were able to predict new motor proteins that the UniProt pipeline was unable to. Therefore, our method is able to help biologist to discover new motor proteins with specific molecular functions accurately. Biologists or scientists can then use this information to recognize the molecular function of motor proteins and help them to design the molecules for...
drug targets in the future.

4. Conclusion

Based on the outstanding results in numerous fields with deep learning, applying it to bioinformatics and computational biology is an important concern for biological researchers. In this study, we have proposed a deep learning technique for classifying the molecular functions of cytoskeleton motor proteins. We have shown that 2D CNN generated from PSSM profiles are able to accurately classify the molecular functions of cytoskeleton motor proteins in organisms (e.g., Homo sapiens and Arabidopsis thaliana). Our method showed a 5-fold cross validation accuracy of 92.6%, 92%, and 90.6% for classifying dynein, kinesin, and myosin, respectively. The accuracy of the independent datasets is 97.5%, 96.4%, and 96.1% respectively. Our approach achieved an improvement across most of the evaluated metrics studied as compared with traditional machine learning techniques. In this study, we proposed a robust model for discovering new proteins that belong to motor superfamilies with high accuracy which can be used to design drug targets. The contributions of this study could provide a basis for further research that can address various bioinformatics problems in the future.

Our contributions take this research a step further and open the doors for further research that will enrich the computational biology field. However, it still bears some limitations and there remain some possible approaches for improving this type of problem in the future.

For instance, the knowledge of the “binding pockets” of proteins with their ligands is the most useful footing for designing therapeutic drugs [79,80] particularly for conducting the mutagenesis studies [79–81]. Therefore, in the future works, there is a need to provide such information about the “binding pocket” and it will help biologists and physicians design the relevant therapeutic drugs. Furthermore, as pointed out in Ref. [82] and demonstrated in a series of recent publications (see, e.g., Refs. [30,35,83,84]), user-friendly and publicly accessible web-servers represent the future direction for developing practically more useful prediction methods and computational tools. Actually, many practically useful web-servers have significantly increased the impacts of bioinformatics on medical science [47], driving medicinal chemistry into an unprecedented revolution [49], we shall make efforts in our future work to provide a web-server for the prediction method presented in this paper.

Table 5
Comparative performance of classifying motor proteins using multi classification and binary classification.

<table>
<thead>
<tr>
<th></th>
<th>Dynein</th>
<th>Kinesin</th>
<th>Myosin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sen</td>
<td>Spe</td>
<td>Acc</td>
</tr>
<tr>
<td>Cross-validation</td>
<td>87.7</td>
<td>90.1</td>
<td>89</td>
</tr>
<tr>
<td>MUL</td>
<td>89.4</td>
<td>95</td>
<td>92.6</td>
</tr>
<tr>
<td>BIN</td>
<td>87.7</td>
<td>93.5</td>
<td>90.8</td>
</tr>
<tr>
<td>Independent</td>
<td>98.3</td>
<td>96.9</td>
<td>97.5</td>
</tr>
</tbody>
</table>


Table 6
Annotation of cytoskeleton motor proteins molecular functions in Homo sapiens (human) and Arabidopsis thaliana.

<table>
<thead>
<tr>
<th>Organism</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Dynein</td>
<td>Kinesin</td>
<td>Myosin</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sen</td>
<td>Spe</td>
<td>Acc</td>
<td>MCC</td>
<td>Sen</td>
</tr>
<tr>
<td>Homo sapiens</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similar vs UniProt</td>
<td>0</td>
<td>7</td>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dissimilar vs UniProt</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arabidopsis thaliana</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Similar vs UniProt</td>
<td>1</td>
<td>55</td>
<td>22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dissimilar vs UniProt</td>
<td>3</td>
<td>3</td>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4. ROC curve for multi-classification of motor proteins. The ROC curve in all three classes almost reached the ideal point (AUC = 0.99, 0.96 and 0.98, respectively). (class 0: dynein proteins, class 1: kinesin proteins, class 2: myosin proteins).


