Design of Software Architecture for Image Tampering Detection

M2019SE004 MIZUTANI Akira
Supervisor : NORO Masami

1 Introduction

Recent technological advances in media tampering has been causes of many harmful forged images. Tamper detection methods[1][2][3] became a major research topic to cope with it in the neural network community. The methods almost always aim at detecting a specific forgeries. That is, a general detecting method to find any tampering has not been invented so far. This paper concerns about a software architecture for organizing multiple neural networks to detect multiple kinds of forgeries. The key issue here is to construct, from the meta-level, a mechanism for an ensemble of front-end neural networks to select a neural network which makes a decision. Under this architecture, we implemented a prototype for detecting forged images resulted from multiple tampering methods of copy-move and compression. In order to demonstrate that our architecture works well, we examined a case study with a total of 120,000 patches which consist of three classes of copy-move, compression and untampered data, 40,000 patches for each. The result shows our proposed method successfully classified 108,954 out of 120,000 patches with 90.82% accuracy.

We also give discussions on our architectural implication to avoid concept drift. Our architecture is designed to be a context-oriented and meta-level, which has a two-layered structure: meta and base. The neural networks can be categorized into base-level components, whereas a component coordinating the networks is addressed in meta-level. The architecture explains that the concept drift can be handled in the meta-level. Through the discussions on the techniques of transfer learning[4], online learning[4], and ensemble learning[4] in terms of the architecture we constructed, it is concluded that we could construct a universal architecture to coordinate machine learning components.

2 Proposed Method

We designed a universal architecture that could be a common base for the cooperation of neural networks. The architecture is context-oriented and meta-level one which gives a structure of two-layered for the sake of dynamic behavior change. We will discuss the design and its details below.

2.1 Design

The architecture we propose is a foundation where neural networks collaborate with one another. The idea is based on the concept of the meta-level architecture reflecting context-orientation. The two-layered structure is required to implement dynamic behavior change.

Since context-oriented structure is necessary when we need to feed the learning result back and use it to make reconfiguration of base-level components, the meta-level architecture is employed. We can put a set of neural networks at the base-level and a component for context and a policy to cooperate the networks at meta-level. Under the architecture, neural networks can be designed to collaborate for handling concept drift.

Techniques handling concept drift such as transfer learning, online learning, and ensemble learning define nothing more than ones which specify meta operation. That is, the techniques show the ways to make neural networks collaborate one another or to change the facilities of neural networks. Both can be regarded as meta operation from the viewpoint of software architecture. In other words, the processes the techniques assume are sequences of meta-level operation and then they can be packed into the policy at meta-level in our architecture. We could also encapsulate the mechanism to change base-level configuration into the policy at meta-level. In addition, the mechanism to change the policy itself as a result of base-level learning, that is, the reflection mechanism, can be also realized with our architecture.

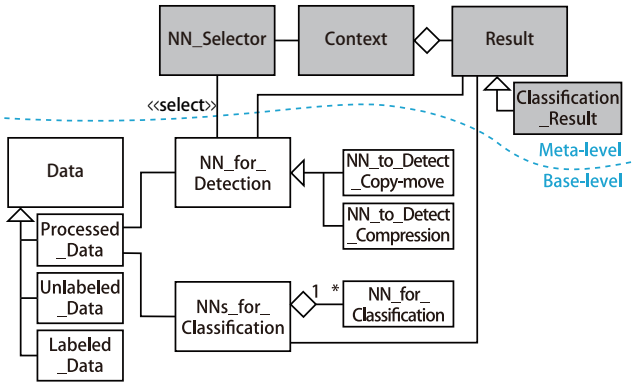
Summing up above, the architecture we propose provides a common base for neural network cooperation including the growth of neural networks. Under the architecture, we can implement techniques advocated in the neural network community to deal with concept drift.

2.2 Details

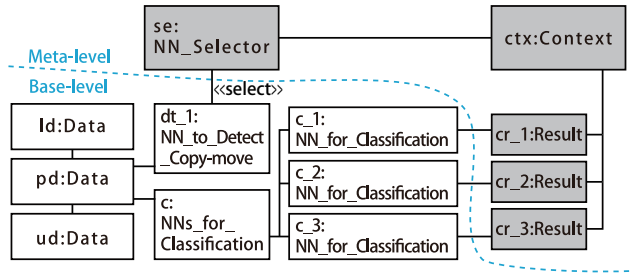
Here, we use an example which is the case study we have done in this research to give details of our architecture. This example is simple enough to describe in the paper and is complex enough to show the advantages of our architecture. This could be without loss of generality since it is fairly useful and practical. It includes enough amount of components and meta-level policy to coordinate base-level neural networks.

UML is used to present the structure and behavior of the architecture. We use class and instance diagram for static structure. An activity diagram is made use of for dynamic behavior. In an activity diagram, a rectangle shows an instance which is referenced somewhere in the diagram. A round rectangle represents a step of a process. We put a message expression in the step.

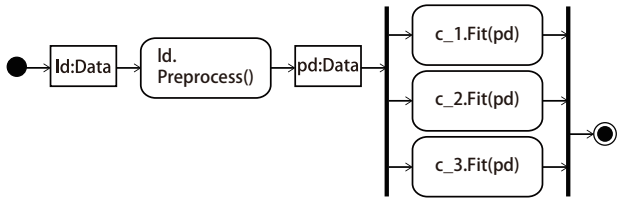
Fig.1 shows an overview of a exampled version of the architecture we propose, (a) and (b) for static structure, and (c) and (d) for dynamic behavior. It is context-oriented as mentioned above, as in (a). Gray-scaled components are meta-level ones coordinating components in the base-level (given as white boxes). The base-level includes a set of *Data*, *NNs_for_Classification*, and *NN_for_Detection*s. Components in the base-level cooperatively work together and make a decision. *NN_Selector* in the meta-level supervise the cooperation. *Data* is a polymorphic type and its instance is preprocessed for the convenience of succeeding procedures. The architecture assumes that *NNs_for_Classification*s are trained for the proper selection of *NN_for_Detection* (see Fig.1 (c)). *NN_for_Detection*s, on the other hand, are assumed to be trained well enough to make an intended decision. In the meta-level, *NN_Selector* accept opinions from *NNs_for_Classification*s, and then, discuss the opinions and finally select a single Detector to make a decision. Thus, base-level components are coordinated by a policy in meta-level components. See Fig.1 (d).



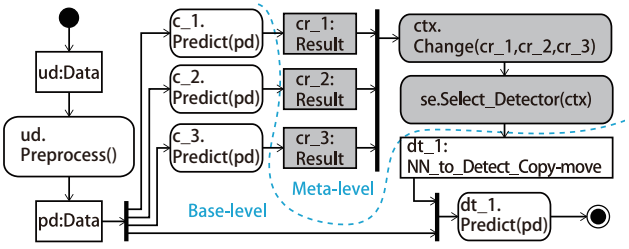
(a) Static structure - Class diagram



(b) Static structure - Instance diagram



(c) Dynamic behavior - Training



(d) Dynamic behavior - Detection

Figure 1: Concrete Architecture

This exampled version is an architecture to deal with concept drift. *NN_Selector* and *Data* are hard-computing components and others are neural networks, that is, soft-computing components. *NN_Selector* standing for neural network selector in the meta-level accepts multiple votes from neural networks for classification. It judges which neural network for detection is the best for finding forgery, and select it from the neural networks for detection (*NN_to_Detect_Copy-Move* and *NN_to_Detect_Compression*). Fig.1 (b) shows a set of instances in case of copy-move forgery suspect. As shown in Fig.1 (d), the selected neural network make a final decision telling if there is a tampering. Hence, through the observation of processes defined in techniques against concept drift, it can be said that we designed the architecture as a natural abstract model for handling concept drift. More details on the components are given below.

Data: *Data* represents a polymorphic type. It has *Preprocessed_Data*, *Unlabeled_Data*, and *Labeled_Data* as its subclasses. *Labeled_Data* is for training and *Unlabeled_Data* for prediction. Both of them are converted into the form convenient for *Classifier* s and *Detector*s when a message for preprocessing is received. In our case study, preprocessings are normalization and high-pass filtering, which are indispensable for the detection of forged images in general.

Neural Network for Classification: The role of the components for classification is to classify the given data into ones tampered by a suspected forgery method. The output of each network is counted as a vote and is sent to *Selector* at meta level.

Multiple classifiers must be prepared since the classification is a cooperative work of learners. In our experiment, three weak learners are implemented for classification. Each network was trained on different pairs of classes. The first one was trained to tell copy-move or compression. The second one classifies copied-moved or untampered data. The last make categorization of compressed or untampered data. Through the experiment, we found that two-out-of-three strategy would work.

Supervised learning is used for the training of the neural networks since predefined sets of tampering detection methods are given.

Neural Network Selector: *NN_Selector* which stands for a neural network selector is a hard-computing component as we assume the selection process is simple enough for a technique such as majority decision. However, depending on the complexity of the task, weighted voting or veto voting may be implemented. Moreover, some cases may require more complicated computation which may be done by a neural network.

Neural Network for Detection: The detection neural networks are a group of candidates which are chosen by *NN_Selector*. Each network is trained to detect a specific forgery. Training of these neural networks is beyond the scope of this study and we assume that there have already been trained networks available.

Selected Neural Network: The selected network will receive the preprocessed input data and return the final decision if the data was tampered or not.

3 Experiment

We conducted a case study to check if our architecture works. Through this experiment, we concluded that our architecture contributes to classify the input data and select an appropriate detector. That is, the purpose of the experiment is the demonstration of the usefulness of our architecture. As we have already shown, Fig.1 outlines our prototype implementation. In this experiment, we implemented the prototype to detect two different forgeries of copy-move and compression. We chose this two since both are typical tampering methods, and there have been already many datasets available for us.

There are the three weak learners for estimating which tampering method is used, and the learned neural networks to check if the given data is forgeries or not in the base-level. The meta-level policy accepts the results from the weak learners, then select an appropriate neural network for detection.

3.1 Training Data

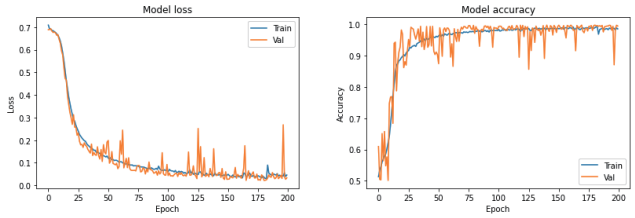
We used MICC-F2000 dataset by Amerini et al[5] for its ease of use and adequate sample size. The dataset is

composed of 700 copy-moved images and 1,300 untampered images.

To simplify our experiment, we decided to classify patch level input instead of the whole image. Thus, we divide our images into patch size of 128*128 pixels. We chose this size as it was the most common among prior researches. For untampered data, we divided untampered images into 128*128 pixel patches. For compression tampered data, we compressed untampered patches to 15% of the original quality. For copy-move data, we sampled patches along the border of the copy-move tampered region. Through these steps, we prepared our 120,000 patches labeled dataset with 40,000 patches belonging to each class.

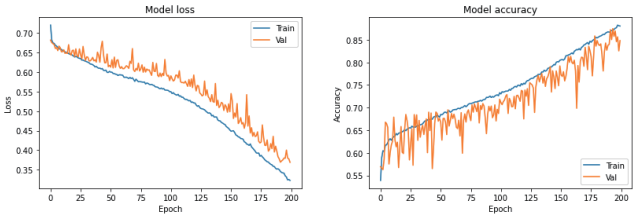
3.2 Results and Evaluation

In the process of learning in our experiment, three weak learners mentioned above are trained. Fig.2, 3, and 4 show learning processes. Although several spikes can be observed, we can assume that learnings are saturated as shapes of graphs present.



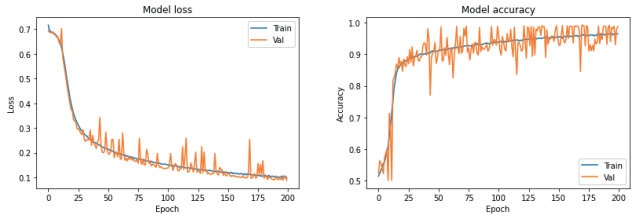
(a) Loss (b) Accuracy

Figure 2: Loss and accuracy of classifier trained with copy-move data and compression data



(a) Loss (b) Accuracy

Figure 3: Loss and accuracy of classifier trained with copy-move data and untampered data



(a) Loss (b) Accuracy

Figure 4: Loss and accuracy of classifier trained with compression data and untampered data

To evaluate our experiment, we calculated the precision, recall, specificity and overall accuracy of our majority voting result. Table 2 shows the precision, recall and specificity of individual classifiers. In this experiment, the overall accuracy turned out to be 90.82%.

With the results in Table 1 and the indices in Table 2, we concluded that our experiment was successful for classifying copy-move, compression and untampered data. The overall accuracy was 90.82% and all of specificity, precision and recall scored above 84%. Although a model more finely tuned and weighted majority decision could improved the result, we did not get further. The reason is that we could assume the result supports usefulness of our architecture. That is, the prototype designed and implemented showed the possibility that the proposed architecture gives a proper structure to choose an appropriate detector out of a set of detectors with weak learner collaboration.

Table 1: Confusion matrix of experiment.

		Prediction		
		Compression	Copy-move	Untampered
label	Compression	39,913	32	29
	Copy-move	26	33,809	6,162
	Untampered	718	4,014	35,232

Table 2: Evaluation for each class

	Compression classification	Copy-move classification	Untampered classification
Specificity	99.07%	94.94%	92.33%
Precision	98.17%	89.31%	85.54%
Recall	99.84%	84.52%	88.15%

4 Discussion

Our architecture is designed to have a context-oriented meta-level structure. In our experiment, we implemented five neural networks at the base-level. Three for classifying the input data by suspected tamper, and the other two for detecting copy-move or compression. The meta-level component coordinates the network to work together and make a selected detector to find a forged image. Collecting votes from the classifiers results in selection of an appropriate tamper detecting method.

We discuss how the architecture handles typical concept drift resolving techniques below. In general, transfer learning, online learning and ensemble learning are said to adapt concept drift. Transfer and online learning are devised to adapt concept drift with the leaning process change. It is a problem on neural network coordination plan in the meta-level of our architecture. Ensemble learning defines the problem as selection and training of cooperative neural networks. Typical techniques of random forest [6] and Adaptable Diversity-based Online Boosting (ADOB) [7] are discussed below. We chose the random forest which is a representative of bootstrap aggregating or bagging, and the ADOB of boosting.

4.1 Transfer Learning

When concept drift occurs, there are usually limited size of data available to train a neural network again. In-

stead of training it from scratch, transfer learning takes the old model and retrains it with the limited new data and change the model into a new version.

In our architecture, the networks before and after transfer learning can be viewed as a base-level component in order to make decision. That is, `Detector` is the base-level component which represents transfer learning. We can omit `Classifiers` from the architecture in this case. The process of the network transfer done by retraining with new data is written in a meta-level component, `Coordinator`. As described above, a beautiful context-oriented implementation of transfer leaning is obtained with the architecture we constructed. The concept is in a context, it has its status indicating if drifted or not, the learning is triggered by context change. We can divide logic for drift check from its action with the architecture.

4.2 Online Learning

Online learning[4] which is a complicated version of Incremental learning[4] overcomes concept drift by means of continuous learning. It shifts the problem of network structure to learning process. That is, it adapts to concept drift by virtue of consistent update of the network with the newest data available. This process can be explained as a meta-level component deciding to train the base-level network with new data.

In our architecture, the meta-level coordinator mixed with the context having training results plays a part of monitoring the learning. A training process proceeds whenever it is required before prediction. The process can be also said that is complicated enough to demonstrate the power of our context-oriented architecture. Again the context has concept as its part, online learning continues according to the context change. Here, logic for drift check is separated from its action with the architecture.

4.3 Ensemble Learning

In general, there are two methods of ensemble leaning: what is called bootstrap aggregated or bugging, and is called boosting. Both achieves its goal of predication with the coordination of neural networks. The bootstrap aggregating controls learning in parallel. The boosting, on the other hand, defines its learning process as a series of learning activities. We chose typical examples for ensemble leaning as mentioned earlier, the random forest and ADOB which represent bootstrap aggregating and boosting respectively.

4.3.1 Random Forest: Bootstrap Aggregating of Ensemble Learning

Random forest includes a software majority circuit to make a decision. In our architecture, a meta-level component, `Coordinator` implements the majority rule. A decision tree corresponds to `Classifier` and no detector is deployed to realize random forest.

Random forest is usually used to provide a simple algorithm such as a majority decision. Our context-oriented architecture helps to write the algorithm as meta-level policy which make cooperation of decision trees, which are weak learners, for example, in our case study above.

4.3.2 ADOB: Boosting of Ensemble Learning

ADOB such as the one used in [7] gives an algorithm for weak learner coordination. The coordinator at meta-level in our network make cooperation of the learners. On the other hand, the weak learners at base-level are repeatedly trained until the best candidate for prediction is designated. As well as bootstrap aggregating, our architecture is also gives a sound context-oriented implementation of boosting, as in this example.

5 Conclusions

We have constructed a context-oriented meta-level architecture for the common base for coordinating neural networks. It gives the structure where a coordinating component in the meta-level monitors the cooperation of components in the base-level. In order to demonstrate the usefulness of the architecture, we applied it to implement an orchestrated set of neural networks to detect tampering of copy-move and compression. We coould achieve a classification at accuracy of 90.82% for 120,000 patches (40,000 each for copy-move, compression, and untampered). We also discussed if our architecture could explain the techniques against concept drift. Through the design, the demonstration, and the discussions, we can conclude that we constructed a universal architecture for neural network coordination.

References

[1] Y. Rao, J. Ni, "A deep learning approach to detection of splicing and copy-move forgeries in images," 2016 IEEE Int. WS Info. Forensics and Security (WIFS), 2016, pp. 1-6.

[2] D. Cozzolino, G. Poggi, L. Verdoliva, "Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection," Proc. 5th ACM WS Info. Hiding and Multimedia Security, 2017, pp. 159-164.

[3] J. H. Bappy, A. K. Roy-Chowdhury, J. Bunk, at al.: "Exploiting spatial structure for localizing manipulated image regions," Proc. IEEE int. conf. computer vision, 2017, pp. 4970-4979.

[4] J. Gama, I. žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," ACM computing surveys (CSUR), 46(4), 2014, pp. 1-37.

[5] I. Amerini, L. Ballan, R. Caldelli, et al.: "A sift-based forensic method for copy-move attack detection and transformation recovery," IEEE trans. info. forensics and security, 6(3), 2011, pp. 1099-1110.

[6] L. Breiman, "Random forests," Machine learning, 2001, 45(1), pp. 5-32.

[7] S. G. T. de Carvalho Santos, R. M. G, Júnior, G. D. dos Santos Silva, et al.: "Speeding up recovery from concept drifts," Joint Euro. Conf. Machine Learning and Knowledge Discovery in Databases, 2014, pp.179-194.