Assignment 3: Deep Learning and Natural Language Processing

Claire Boetticher, MSDS 458, SEC 56

Colab: https://colab.research.google.com/drive/1w8O2d6qmNGTE0QOh31Zbw2NHRY7OmLVF?usp=sharing

Abstract

Natural language processing (NLP) entails the design, development, and application of computational algorithms to automatically analyze and represent human language. Traditionally, shallow machine learning models were applied, requiring resource-intensive manual efforts in pre-processing and feature engineering. Recent successes in utilizing deep neural networks have provided powerful advancements in a variety of challenging NLP problems, building on advances in embeddings for text representation that better address the challenges posed by high-dimensionality in text vectors. Dense neural networks (DNNs), recurrent neural networks (RNNs), and also convolutional neural networks (CNNs, traditionally used with image data) all offer potential paths forward with capturing meaning from textual data and enabling powerful applications such as machine translation, speech recognition, automatic summarization, and question answering. Though deep learning is neither a silver bullet for NLP tasks nor an approach that truly understands text as humans do, these kinds of models do offer new methods for mapping the statistical structure of written text through powerful pattern recognition.

Introduction

This study is a comparative exploration of dense, recurrent, and convolutional neural networks for news classification with the Reuters newswire dataset, containing 11,228 newswires from Reuters labeled over 46 topics. This well-known text dataset is useful for experimenting with models to achieve reliable discriminatory power between topics. The study has a two-fold goal: develop a neural network model for highly-accurate and generalizable classification and gain an in-depth understanding of how various factors affect fitting and ultimate test set performance with networks of differing topologies and hyperparameter settings. This will shed light on how hidden nodes learn to extract features from inputs; with additional layers, the objective is to better understand how each successive layer extracts progressively abstract and generalized features. Specifically, the study will accomplish this through exploration of various neural network structures and topologies, including DNNs, RNNs with and without Long Short-Term Memory (LSTM), and one-dimensional CNNs. Overfitting and the challenge of vanishing gradients, in addition to the so-called curse of dimensionality, are a constant presence with training these models and attempting to design for generalizability. Various mitigating steps, for example using LSTM RNNs and regularization techniques, will be evaluated for impact on performance and future utility on unseen data.

Literature review

With numerous developments in deep learning in the NLP space, made possible by computing advancements and the vast availability of training data, there seems to be a recent (past five years or so) shift toward more challenging areas of human language analysis, such as developing models that represent meaning and context. Word embeddings (also called distributional vectors), embedding syntactical and semantic information from human language, have gained popularity with the continuous bag-of-words (CBOW) and skip-gram models from Google (Mikolav et al., 2013). The

GloVe model from Stanford University represents a significant advancement in embeddings as well, made openly available for a variety of NLP tasks from sentiment analysis to classification (Pennington et al., 2014).

Machine translation, another area building on momentum from advances in sequential processing, is growing in popularity as a research area. The sequence-to-sequence model by Sutskever et al. (2014), which Google Translate employs, and the transformer architecture, which utilizes an attention mechanism that assesses an input sequence then decides at each step which other parts of the sequence are meaningful, made significant translation advances possible (Vaswani et al., 2017). A final compelling area of research seeing growth is ethics in NLP, with a focus on gender, racial, and ethnic biases examined in certain pre-trained word embeddings. The implications for the biases "learned" from the various massive text collections used to develop embeddings, as a result, impact the NLP applications reliant on those embeddings (Kaneko and Bollegala, 2019; Zmigrod et al., 2019; Stanovsky et al., 2019).

Methods

The Reuters dataset contains 11,228 newswires from Reuters labeled over 46 topics that span a variety of news subjects, including trade, reserves, jobs, and also certain commodity labels like crude, iron-steel, gold, and wheat. The label distribution across the classes varies quite a bit, with some topic labels occurring too infrequently to suggest reliable discriminatory power of any classification model. Exploratory analysis reveals that the top nine most frequent topics match between training and test datasets, leading to the approach to address the imbalance by reducing the datasets to only those newswires whose labels match the top nine (money-fx, trade, ship, interest, crude, earn, money-supply, acq, grain). The reduced dataset is divided into 7503 newswires for training (with 15 percent held back for validation) and 1852 newswires for testing. Text sequences in Experiment 1 are converted to 10,000 length vectors as the expected network input using one hot coding. Embeddings are used in other experiments to determine effect on model performance with denser, lower-dimensional text representations. Embeddings' input length are set at 193, the 80th percentile max length of training data. The study consists of four separate experiments, with varying neural networks evaluated for performance (measured by classification accuracy) and processing time. Architectures are defined as follows:

Type of Neural Net	Network Architecture	Trainable Parameters
DNN 1.1	2 hidden layers (64 / 32 nodes)	642,441
DNN 1.2	2 hidden layers (256 / 256 nodes)	2,628,361
DNN 1.3 (dropout)	2 hidden layers (64 / 32 nodes)	624,441
DNN 1.4 (dropout + L1)	2 hidden layers (64 / 32 nodes)	642,441
DNN 1.5	1 embedding layer (input length = 193) 2 hidden layers (64 / 32 nodes)	2,863,497
DNN 1.6 (dropout)	1 embedding layer (input length = 193) 2 hidden layers (64 / 32 nodes)	2,863,497
RNN 2.1	1 embedding layer (input length = 193) 64 memory cell units	1,293,717

RNN 2.2	1 embedding layer (input length = 193) 100 memory cell units	1,305,021
LSTM RNN 3.1	1 embedding layer (input length = 193) 32 memory cell units	653,109
LSTM RNN 3.2	1 embedding layer (input length = 193) 100 memory cell units	708,121
LSTM RNN 3.3 (dropout)	1 embedding layer (input length = 193) 100 memory cell units	708,121
LSTM RNN 3.4 (variational)	1 embedding layer (input length = 193) 100 memory cell units	708,121
CNN 4.1	1 embedding layer (input length = 193) 1 convolutional layer (32 filters)	3,786,805
CNN 4.2 (dropout)	1 embedding layer (input length = 193) 1 convolutional layer (32 filters)	3,786,805
CNN 4.3 (dropout)	1 embedding layer (input length = 193) 1 convolutional layer (32 filters) 1 max pooling layer	2,213,941

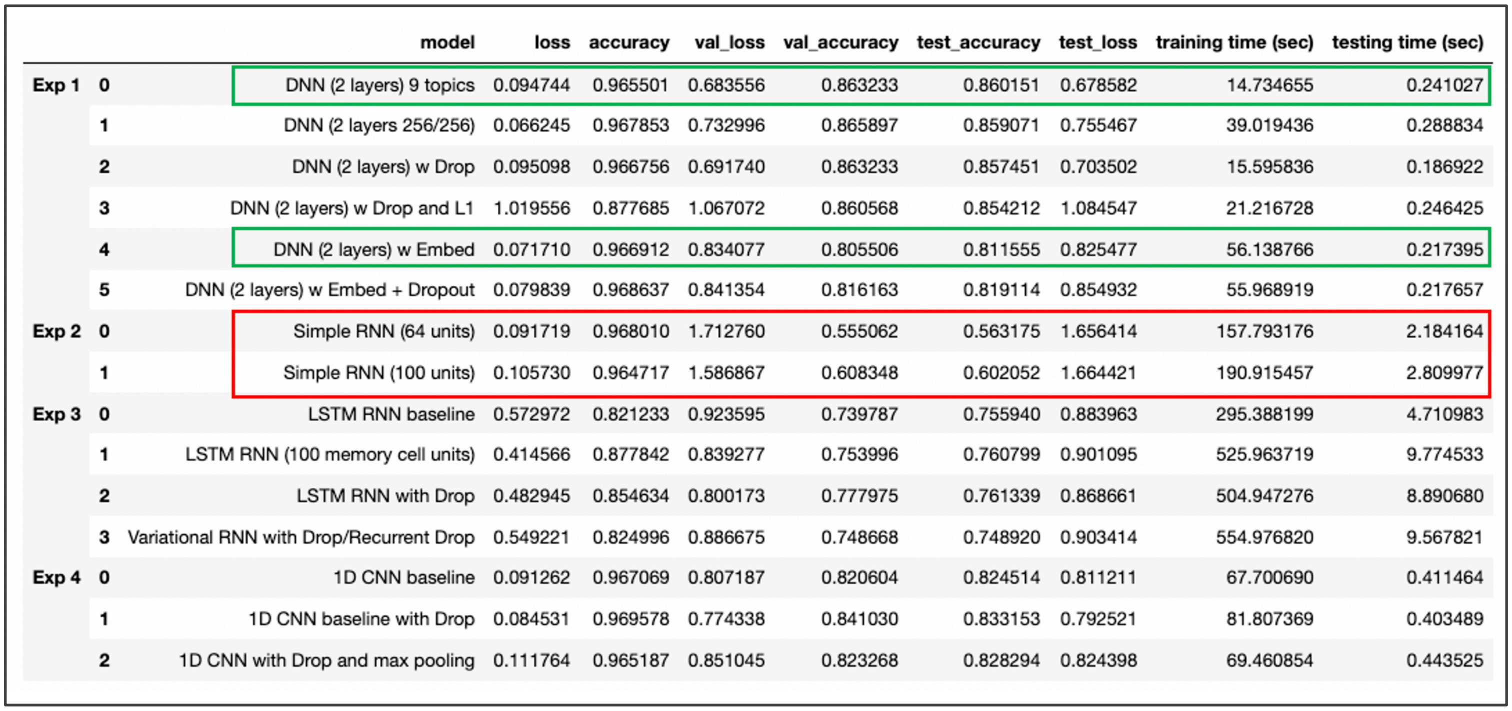
Experimental networks contain varying neural layers, all with softmax activation in the last layer of the probability distribution over the nine different output classes (topic labels). Dense neural network 1.1 serves as a baseline model with two hidden layers. Subsequent DNNs include additional hidden layer units in each of the two layers (1.2), dropout regularization (1.3), dropout and L1 regularization (1.4), the addition of a learned embedding layer (1.5), and embedding plus dropout regularization (1.6). Embedding layers are included in certain experiments as a potential means to address the sparsity and high-dimensionality of the input vectors, both a product of the 10,000-word vocabulary used in this study. Employing L1 regularization, which constrains the original loss function, enables an examination of effects on model fit, classification performance, and process time. Given the potential of overfitting with the networks under examination and the increasing number of tunable parameters with some, attention is given to that scenario in model performance with both training and validation data. Simple RNNs in Experiment 2 consist of a baseline with 64 memory cell units (2.1) and one with 100 memory cell units (2.2). Experiment 3 employs LSTM RNNs, a variant intended to address the limitations imposed by the vanishing gradient problem. These models consist of a baseline model with 32 memory cell units (3.1), a wider LSTM layer with 100 memory cell units (3.2), and the addition of dropout regularization (3.3). Additionally, a Variational RNN is tested to evaluate the potential impact of using the same dropout mask at each time step, including the recurrent layers (3.4). Experiment 4 evaluates one-dimensional CNNs with a single convolutional layer (4.1), the addition of dropout regularization (4.2) and the addition of a max pooling layer (4.3). The following hyperparameters are employed across all experiments:

- Batch size: 100 / Epochs: 10
- Activation: ReLU
- Optimizer: Adam

- Loss function: Categorical cross-entropy

Results

The table below shows model fitting and evaluation results :



		model	loss	accuracy	val_loss	val_accuracy	test_accuracy	test_loss	training time (sec)	testing time (sec)
Exp 1	0	DNN (2 layers) 9 topics	0.094744	0.965501	0.683556	0.863233	0.860151	0.678582	14.734655	0.241027
	1	DNN (2 layers 256/256)	0.066245	0.967853	0.732996	0.865897	0.859071	0.755467	39.019436	0.288834
	2	DNN (2 layers) w Drop	0.095098	0.966756	0.691740	0.863233	0.857451	0.703502	15.595836	0.186922
	3	DNN (2 layers) w Drop and L1	1.019556	0.877685	1.067072	0.860568	0.854212	1.084547	21.216728	0.246425
	4	DNN (2 layers) w Embed	0.071710	0.966912	0.834077	0.805506	0.811555	0.825477	56.138766	0.217395
	5	DNN (2 layers) w Embed + Dropout	0.079839	0.968637	0.841354	0.816163	0.819114	0.854932	55.968919	0.217657
Exp 2	0	Simple RNN (64 units)	0.091719	0.968010	1.712760	0.555062	0.563175	1.656414	157.793176	2.184164
	1	Simple RNN (100 units)	0.105730	0.964717	1.586867	0.608348	0.602052	1.664421	190.915457	2.809977
Exp 3	0	LSTM RNN baseline	0.572972	0.821233	0.923595	0.739787	0.755940	0.883963	295.388199	4.710983
	1	LSTM RNN (100 memory cell units)	0.414566	0.877842	0.839277	0.753996	0.760799	0.901095	525.963719	9.774533
	2	LSTM RNN with Drop	0.482945	0.854634	0.800173	0.777975	0.761339	0.868661	504.947276	8.890680
	3	Variational RNN with Drop/Recurrent Drop	0.549221	0.824996	0.886675	0.748668	0.748920	0.903414	554.976820	9.567821
Exp 4	0	1D CNN baseline	0.091262	0.967069	0.807187	0.820604	0.824514	0.811211	67.700690	0.411464
	1	1D CNN baseline with Drop	0.084531	0.969578	0.774338	0.841030	0.833153	0.792521	81.807369	0.403489
	2	1D CNN with Drop and max pooling	0.111764	0.965187	0.851045	0.823268	0.828294	0.824398	69.460854	0.443525

The simple baseline DNN with reduced topics (Model 1.1) performs best overall across metrics of interest (accuracy, loss, and process time) (Figure 1). Model 1.2 with wider network layers (256 units) has comparable training and validation performance to Model 1.1, with slightly lower training loss; this does not suggest that the added process time and complexity is necessarily warranted (Figure 2). Dropout and L1 regularization do not seem to contribute to performance in a notable way in Models 1.3 and 1.4, though learning curves converge and trend in a way that suggests overfitting may be mitigated (Figures 3 and 4). Model 1.5 with the embedding layer performs well, with slightly lower training loss than the other models in Experiment 1 but slightly higher validation and test loss than Model 1.1 without an embedding layer, suggesting that that layer may not necessarily contribute to better performance on unseen data. Dropout regularization in Model 1.6 does not seem to contribute in the way of generalizability, with validation loss still high and seeming to climb as epochs progress (Figure 6).

Baseline RNN 2.1 in Experiment 2 performs very well on training, but quite poorly on validation and test data. Test accuracy is middling and validation and test loss are so high as to suggest overfitting and very unreliable performance on unseen data (Figure 7). Adding model width by increasing memory cell units to 100 in RNN 2.2 does not increase performance and still results in very high validation and test loss (Figure 8). Process time and validation and test loss all increase notably with RNN model fitting in both treatments.

In Experiment 3 with LSTM RNNs, training accuracy is high and comparable to simple RNNs, but Model 3.1 performs much better on validation data. Training and validation loss are lower as well; generally, loss and accuracy curves converge and trend in a more promising direction (Figure 9). Test accuracy is higher and test loss is lower as well, suggesting that the LSTM RNN may generalize more reliably, depending on the data. Model 3.2, with a wider LSTM layer with 100 memory cell units (versus 32 in Model 3.1), performs slightly better on training and validation data but has a higher test loss value; the loss and accuracy curves converge and trend in promising directions again, like Model 3.1 (Figure 10). Training and validation accuracy were higher in Model 3.3, applying ten percent dropout regularization to the 32 outputs from the LSTM layer as input to the Dense layer (Figure 11). Model 3.3 training and validation loss were lower as well, with test performance comparable to Model 3.2. The Variational RNN in Model 3.4 does not contribute notable performance improvements for training, validation, or test data (Figure 12), suggesting this model architecture may not necessarily be appropriate for this dataset or task. Process time for LSTM RNNs is exponentially higher than previous networks, doubling for the baseline in Model 3.1 and five times longer for the networks with 100 memory cell units.

The baseline 1-D CNN performs well on training and validation data, though validation loss is still relatively high and unpredictable within ten epochs (Figure 13). The model performs well on test data, suggesting more potential for generalizability. Model 4.2, which applies 50 percent dropout regularization before the last Dense layer, performs slightly better than 4.1 in terms of loss and accuracy for validation data (Figure 14). Test performance is slightly better than Model 4.1. Loss values across training, validation, and test data are lower in Model 4.2 than all others from this experiment. Model 4.3, with both convolutional and max pooling layers and dropout regularization, does not perform noticeably better than the other models in this experiment in terms of train, validation, and test accuracy; loss values are slightly higher than Models 4.1 and 4.2 (Figure 15). The CNNs from Experiment 4 all have much faster process times than the RNNs (simple and LSTM).

Conclusions

The performance and simplicity of the dense neural networks in this study suggest that the nature of the Reuters dataset and the task, as defined in this study, may not warrant the complexity or sophistication of RNNs or CNNs. In fact, deep learning approaches may not be necessarily at all, given the size and nature of this data. The choppiness and short-hand prevalent throughout the newswires suggest an alternate approach using shallow classification algorithms that focus more on single word contribution to “meaning” without the sequential dependence may be more appropriate. The reduced topics also focus heavily on financial topics (for example, money, ear, interest). If these topics’ language contain terms that cross over frequently across labeled newswires, it could have had an effect on the models’ discriminatory capabilities.

The experiments do shed light on some interesting effects of certain treatments, though, and enable speculation on why this dataset may not have seen notable performance improvements with the increasing sophistication of some of the methods applied. Word embeddings, for one, do offer density, lower-dimensionality, and in some cases, an enhanced capturing of relationships amongst words in sequences and across entire documents and corpora. They are not, however, a silver bullet: the Reuters dataset, and the nature of the newswires within, may not lend themselves to the manner in which embeddings learn formal semantics and the constructed models underlying those embeddings. The tradeoff between the understanding expected from embeddings (which may be worthwhile with a subject-specific document or a type of document for which pre-trained embeddings may be appropriate, like formal news reporting) on one hand and the added parameters and complexity on the other hand to consider. As for RNNs, while they are designed to learn and generalize across sequences of inputs, there is no guarantee of performance across all NLP tasks.

Specific to this study, the SimpleRNN Keras layer is not well-equipped to learn long-term dependencies amongst words and thus process long sequences from text. This study's design implements this layer rather simply and without much hyperparameter tuning, which could contribute to the poor performance in addition to the vanishing gradient problem at hand. The LSTM RNNs in Experiment 3 do seem to perform much better than those networks trained with SimpleRNN layers; importantly, their learning curves also converge and trend in a manner more suggestive of progressive improvement over Experiment 2's simple RNNs. However, the performance gains are mostly limited to training data; the high loss values in validation and test data all raise a red flag on potential to generalize. Finally, the one-dimensional CNNs offer some performance improvement over LSTM RNNs (training, validation, and testing) at a fraction of the process time – one key advantage of CNNs being their efficiency and speed. Loss values are still quite high for validation and testing data, though, again raising a concern that these models have "learned" the training data well but do not generalize reliably. Additionally, the plateau seen in training and validation accuracy may suggest that the dataset is too small and limited to merit (and make use of) the CNN's sophistication. This study provides multiple starting points for further development, though. Ideally, if more data were available, the existing models could be improved. Otherwise, these baselines could serve as a point to explore further hyperparameter tuning and even the use of pre-trained embeddings (ones trained on news). Further text pre-processing could be incorporated as well to test impact on performance and generalizability.

This study ultimately raises a number of important points worth considering in applying deep learning models to an application such as a machine learning-enabled conversational agent or chatbot to assist customer support representatives. Management would be well advised to consider, first and foremost, the availability of a large and diverse training dataset that encompasses not only the important words that will likely come up in conversations (e.g., product names, expressions of sentiment, synonyms) but also the conversational element of customer support. Ideally, written transcripts representative of a variety of previous support calls would be a good start. Assuming an understanding of customer expectations and substantive data available for training models to deduce meaning from this data and then use it make reliable predictions about what a customer may want or ask for next, models worth testing would likely include RNNs of some sort, particularly LSTM RNNs that could potentially extract and save relevant information across sequences, preventing older signals from disappearing over training. This design consideration is critical considering the progressive and additive nature of customer support: a customer may ask an initial question but the conversation will develop and evolve (if they are engaged enough to continue using, of course) and take unexpected directions. Models need to account for this kind of variety and the potential connection between all intervals of information from beginning to end. Again, a large variety of conversation transcripts covering the range of tasks a customer may wish to complete will aid this training process. This would likely be a substantial development project - time-consuming and potentially expensive given the computational power needed for models like RNNs - even with data already available for training. Multiple baselines would need to be developed across model types, with single and ensemble models and extensive hyperparameter tuning. Learned embedding layers may help in representing the unique nature of a given company's product line and customer base, and how these conversations reflect that. From a user experience perspective, the model outputs as designed into an agent workflow would need to be tested extensively with feedback incorporated prior to deployment. After deployment, even with models considered reliable and fit for production, regular training will likely be required to ensure model relevance as assumptions are proven wrong, product names or qualities change, customer bases expand and evolve, and other considerations that developers might not have expected.

References

Kaneko, M. and Bollegala, D. (2019). Gender-preserving Debiasing for Pre-trained Word Embeddings. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality.

Pennington, J., Socher, C., Mannin, D. (2014). GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

Stanovsky, G., Smith, N., Zettlemoyer, L. (2019). Evaluating Gender Bias in Machine Translation. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems — vol. 2 (NIPS'14).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2017). Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17).

Zmigrod, R., Mielke, S., Wallach, H., Cotterell, R. (2019). Counterfactual Data Augmentation for Mitigating Gender Stereotypes in Languages with Rich Morphology. Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics.

Appendix

Figure 1: Learning curves for Model 1.1

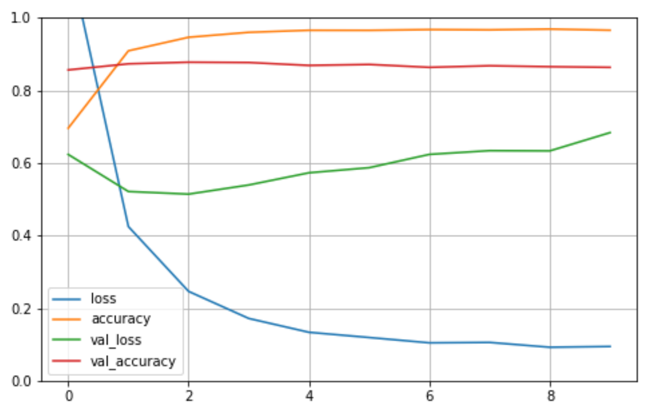


Figure 2: Learning curves for Model 1.2

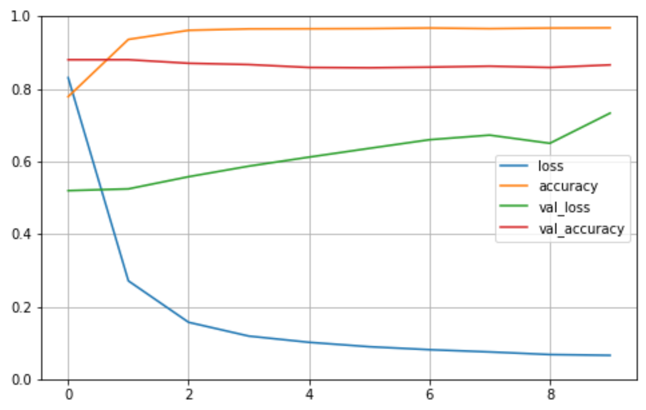


Figure 3: Learning curves for Model 1.3

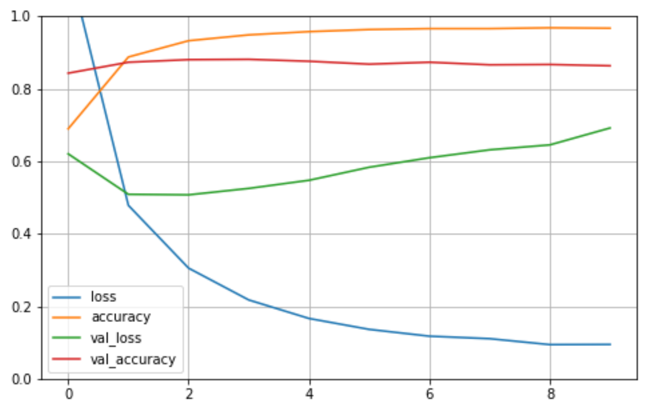


Figure 4: Learning curves for Model 1.4

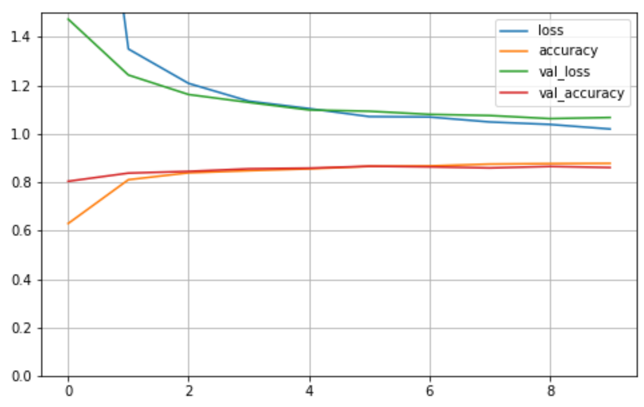


Figure 5: Learning curves for Model 1.5

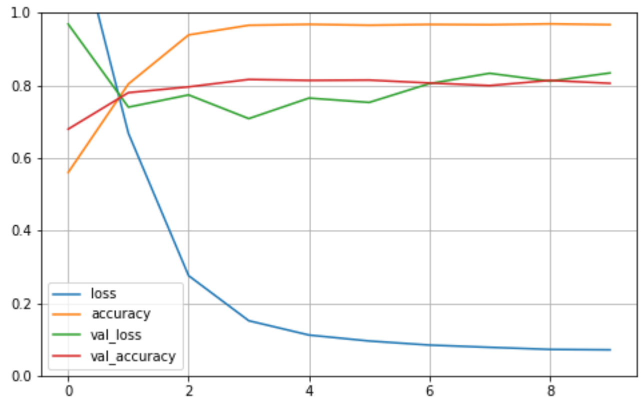


Figure 6: Learning curves for Model 1.6

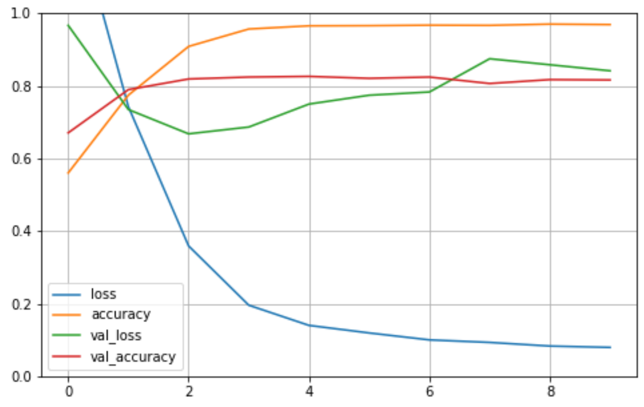


Figure 7: Learning curves for Model 2.1

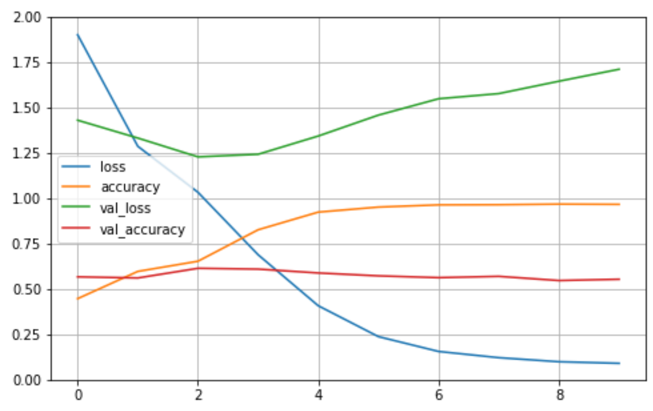


Figure 8: Learning curves for Model 2.2

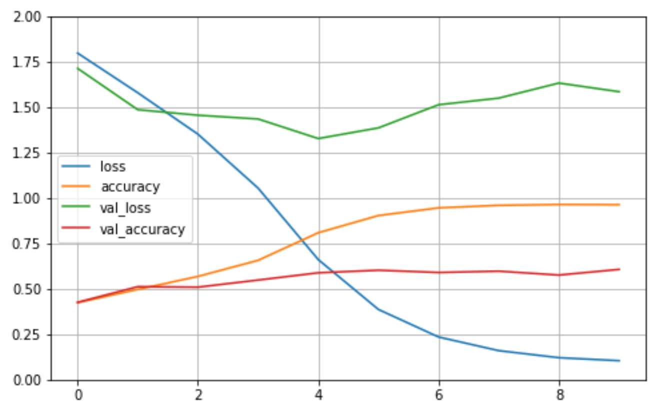


Figure 9: Learning curves for Model 3.1

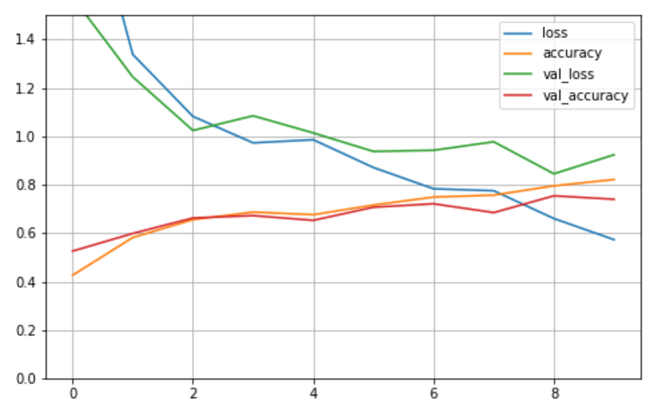


Figure 10: Learning curves for Model 3.2

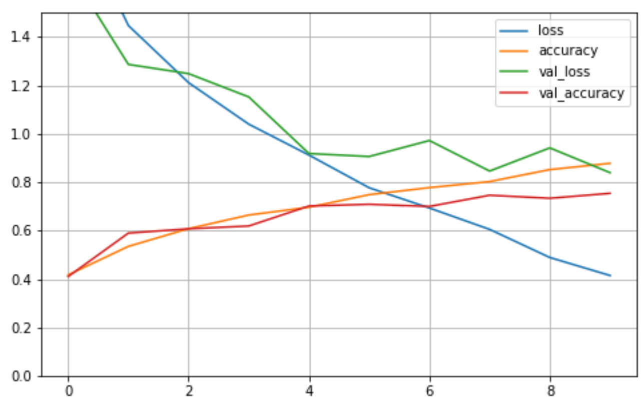


Figure 11: Learning curves for Model 3.3

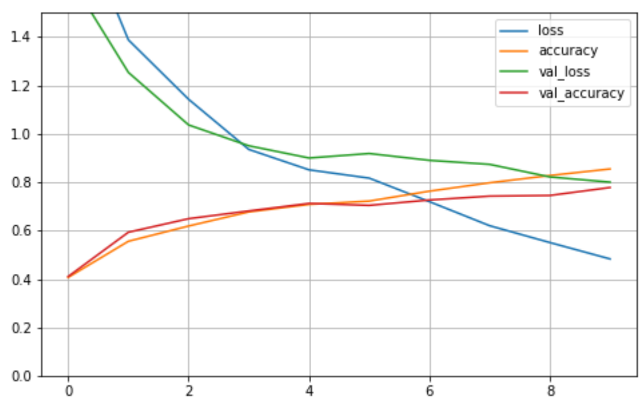


Figure 12: Learning curves for Model 3.4

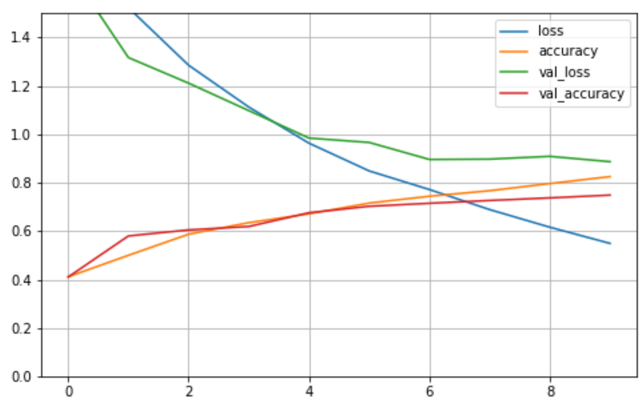


Figure 13: Learning curves for Model 4.1

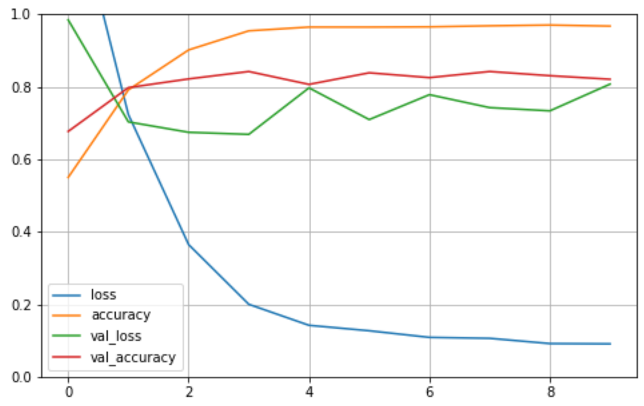


Figure 14: Learning curves for Model 4.2

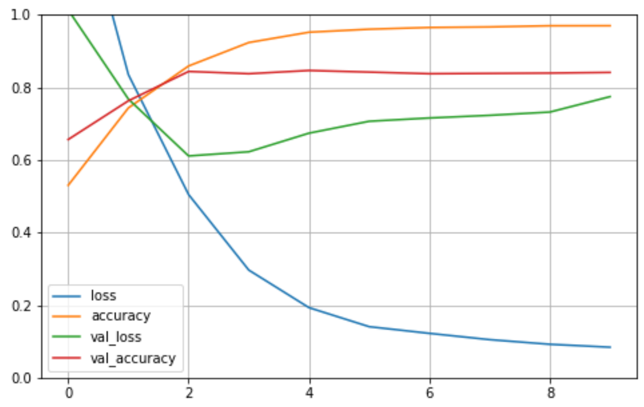
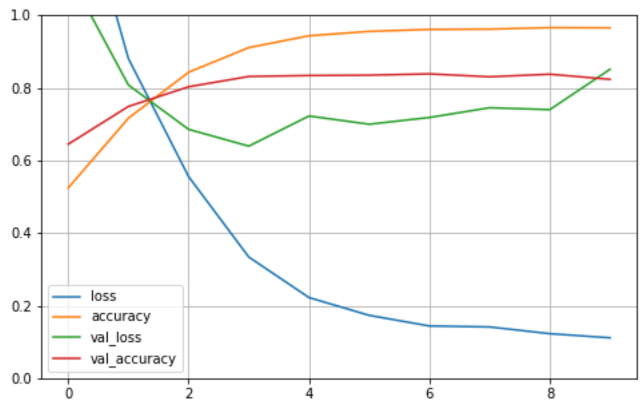


Figure 15: Learning curves for Model 4.3


loss
accuracy
val_loss
val_accuracy