# A Deep Learning Decoder for Long-Range Communication Systems

Damián Pascual, Simon Tanner, Mickey Vänskä, Roger Wattenhofer
*ETH Zurich*
Switzerland
{dpascual, simtanner, mickeyv, wattenhofer}@ethz.ch

***Abstract*—Long-range communication systems require receivers that can detect and decode messages in spite of strong distorting effects. However, classical decoders often fail at coping with complex effects such as interference or multipath propagation. Deep learning has shown strong generalization and adaptation capabilities and is a promising approach for improving decoding systems. In this work, we study the specific case of aircraft communication and build a purely deep learning-based receiver. It detects incoming messages, finds the exact starting point and then decodes their message bits. We demonstrate the performance of our system and show that it can decode 45 % more messages than a classical baseline decoder. Our approach is general and can be adapted to many communication protocols.**

***Index Terms*—Message detection, aircraft communication, deep learning**

## I. Introduction

Receivers capable of detecting and decoding messages are one of the fundamental building blocks of any communication system. However, the task of receiving a signal and decoding its content can be challenging due to interference, multipath propagation and other distortions. The systems are also often based on assumptions about the exact implementation of the transmitter hardware and the propagation path. Despite modern decoding systems becoming more robust against such effects, there is still a gap that may be bridged by designing deep learning-based decoders. By using deep learning, we can train a receiver to robustly detect and decode messages that suffer from strong distorting effects using labeled data from receivers in multiple locations. We simultaneously use multiple conventional receivers to collect training data. In this way we train a single deep learning-based receiver which then achieves better performance than the classical receiver used as baseline.

In this work, we apply this idea to messages transmitted by aircraft. Aircraft regularly broadcast their position, velocity, and other information. Nearby ground receivers and other aircraft receive these messages and use them for situational awareness. This communication is called *ADS-B* (Automatic Dependent Surveillance - Broadcast), messages are sent in the Mode S channel at 1090 MHz.

ADS-B capable transponders are mandatory in the USA since January 2020 in most controlled airspace [1]. Similarly, these transponders will also be mandatory in the European Union for large aircraft from June 2020 [2].

The authors of this paper are alphabetically ordered.

ADS-B messages can also be received using software-defined radios (SDR). Various websites exist that combine the information from many simple SDRs to track aircraft, such as FlightAware and Flightradar24. Most of these projects rely on the open-source dump1090 decoder software to detect and interpret the ADS-B messages.

Decoders such as dump1090 [3] are software decoders specifically designed and optimized to detect these messages using state of the art signal processing techniques. But, are there ways to detect messages more reliably? In this work, we answer this question by using deep learning to detect and decode ADS-B messages.

Our receiver consists of three neural networks. The first determines the existence of a message in a window of samples from the SDR. Then, the exact starting point is determined. Finally the individual bits contained in the message are decoded.

We use messages collected simultaneously at different locations to generate additional training data consisting of messages that cannot be locally decoded by dump1090. We evaluate two different training regimes of our system; one with exclusively locally decodable messages and one including locally non-decodable messages. Our results show that a decoder based on deep learning can outperform classical decoders. Given the generality of our approach, our system can be successfully implemented for many other communication protocols.

## II. Related Work

Deep learning [4] has shown outstanding performance and strong generalization in very complex tasks such as image classification [5] and generation [6], natural language processing [7] or audio enhancement [8].

Recently, deep learning has been adopted to improve parts of communication systems such as modulation recognition, channel estimation, etc. [9]. Deep learning has also been used to decode codes, such as linear codes [10] and polar codes [11]. Ye et al. [12] show how deep learning can be used in OFDM systems to estimate the channel and recover the transmitted symbols. Our work shows how deep learning can be used for the complete detection and decoding of ADS-B messages.

In the context of ADS-B, deep learning has been previously used to detect anomalous messages by observing sequences of messages [13]. Also an intrusion detection system based on received signal strength (RSS) patterns has been proposed [14]. As ADS-B does not provide any message authentication, identifying the transmitting aircraft based on physical or software-dependent features of the signal is also an active research topic [15].

Autoencoders have been investigated for wireless communication and show promising results also in MIMO systems [16], [17]. In this approach, a model comprising both sender and receiver is trained simultaneously. Therefore, also the encoding is learnt and does not correspond to an already existing protocol. However, in our case we aim at decoding messages sent with a given protocol (ADS-B) and hence, we can only optimize the decoding system and we do not adapt the encoding.

## III. Background

ADS-B (Automatic Dependent Surveillance - Broadcast) is a protocol used for air traffic surveillance. The aircraft regularly broadcasts its position, which is determined using satellite navigation, to nearby aircraft and ground stations. Additionally, velocity and other information is broadcast. Each message also contains a 24-bit field that uniquely identifies the aircraft.

Mode S has been developed as a secondary surveillance radar technique. The aircraft responds at $1090\,\mathrm{MHz}$ to an interrogation sent from a ground station at $1030\,\mathrm{MHz}$. Two different lengths of Mode S messages exist with 56 or 112 bits. Under good conditions, Mode S messages can travel hundreds of kilometers as they are sent with a power of up to $500\,\mathrm{W}$. The information is modulated using pulse-position modulation (PPM) with a symbol length of $1\,\mu\mathrm{s}$. Additionally each transmission starts with a fixed preamble of $8\,\mu\mathrm{s}$ containing four pulses.

The ADS-B messages are transmitted over the Mode S extended squitter link where every message contains 112 bits. These messages are regularly broadcast by the aircraft without interrogation from the ground. For instance, messages containing the position of the aircraft are sent twice a second.

Since we are interested in receiving ADS-B messages, we focus on Mode S messages with downlink format 17. ADS-B messages are contained in these Mode S transmissions and contain a 24-bit CRC with a Hamming distance of six. This CRC allows the reliable correction of two bit errors or the identification of messages with up to five errors [18]. Therefore, the goal of our deep learning-based receiver is to decode as many messages as possible with two or fewer bit errors.

## IV. Data Collection

To generate the training, validation and test data for our deep learning-based receiver, Mode S messages are collected using an SDR. A USRP B200mini is used to sample the signals at $1090\,\mathrm{MHz}$ at $12\,\mathrm{MS/s}$. This stream is then downsampled to $2.4\,\mathrm{MS/s}$ and fed to dump1090. After a message has been detected and decoded by dump1090, the corresponding samples are cut out from the $12\,\mathrm{MS/s}$ stream and saved to a database together with the decoded message bits. Although the message length is $1440$ samples, we save windows of $2700$ samples that completely contain the message with a randomly chosen amount of additional samples before it. This way, we can train our system to detect messages irrespective of their position in the window.

With this setup, we can only record messages that also dump1090 can decode. To also record messages that dump1090 has missed due to interference or multipath propagation, we also collect messages simultaneously at other locations. This allows us to exceed the performance of dump1090 with the deep learning approach.

A server collects messages from multiple receivers in a database. The messages are detected from up to eight receivers distributed across the northern part of Switzerland. Each receiver consists of a Raspberry Pi 3 with an attached RTL-SDR and runs dump1090. Based on the measured timestamps of the received messages, the server calculates the time offsets and drifts between the receivers and the exact send timestamps of the messages at the aircraft, similar to [19].

Also the messages detected using the USRP are sent to the server. Therefore, also this receiver takes part in the synchronization and we can calculate when each message detected by the other receivers should have arrived at the USRP receiver.

To exactly find the position of the message in the received signal, we compute the cross-correlation between the signal and the ideal message that we expect. The ideal message is constructed by modulating the message bits received from the server with the ideal pulse shape. The maximal correlation value provides us with a measure of the signal quality for the messages that cannot be decoded locally. Only messages with a correlation value above a certain threshold and arriving close to the expected time at the USRP are accepted. Otherwise, we would give the deep learning methods many labeled messages that are not detectable at the location of the USRP due to strong interference, obstructions or a too large distance. In the following, we call these messages that we find with the help of the distributed receivers ”low quality messages”.

In summary, we collect two kinds of messages, locally decodable by the USRP, or high quality messages; and non-locally decodable, or low quality messages. Furthermore, we collect signals where no message exists that we call ”silence”. For each of these messages we provide three labels: a binary value for the existence of the message in the signal (and if so), starting point within the window, and bits of the decoded message.

## V. Deep Learning Decoder

The pipeline of our decoding system is based on deep neural networks and consists of three stages: message detection, offset calculation and bit decoding. Each of these actions is performed with a specific neural network.

### A. Message detection

Aircraft only send very short messages to limit the chance of colliding with messages from other aircraft. Therefore, to correctly decode a message in the continuous stream of samples, it is necessary to first determine whether a given segment of the signal contains a message or not. For this purpose we use a one-dimensional convolutional neural network (CNN) that takes as an input a segment of 2700 samples and produces a binary output, 0 if there is no message, i.e., silence, and 1 otherwise. If a message is only partly contained in the segment, the segment is also considered as containing no message. This network consists of seven convolutional layers with batch normalization, kernel size of 24 and increasing number of filters (64, 64, 128, 128, 256, 256 and 512). After the convolutions there is a final fully connected layer with sigmoid activation function.

### B. Offset calculation

Once a given segment is identified as containing a message, we need to calculate the actual starting point of the message. Since the signal length we are considering in the detection stage is 2700 samples and the actual length of the message is 1440, the offset of the message starting point can be as large as 1260 samples. In this stage of the pipeline we use a Residual Network (ResNet) [20] with six convolutional layers with residual connections and max pooling, and a final fully connected layer. Again, batch normalization is applied and the kernel size is 24, while the number of filters is 64, 128, 128, 256, 256 and 512. This network takes as input a signal of length 2700 samples and outputs the position in the signal at which the message starts. Using this information, we cut the signal to a length of 1600 samples, giving a slack of 160 extra samples in order to correct for possible inaccuracies in the offset calculation.

### C. Bit decoding

After detecting the existence of a message and cutting the signal to a length of 1600 samples, we pass the resulting segment to the last stage of the pipeline, the bit decoding. In this stage we use a neural network of the same architecture as the message detection network except that it has an eighth convolutional layer with 512 filters. This network is trained to produce at the output a binary vector of length 112 where each element corresponds to a bit of the decoded message.

## VI. Evaluation

To evaluate our decoding system, we compare it to the classical decoder dump1090, which we use as a baseline. We demonstrate the performance of the three stages of our pipeline and show that our deep learning-based receiver is more robust to low quality messages than the baseline.

### A. Pipeline performance

To validate the performance of our pipeline, we first evaluate it in the task of decoding the high quality messages that can be decoded by the baseline decoder. This way, we build training, validation and test sets as explained in Section IV. Each one of these sets is recorded on a different day in order to ensure that the model does not overfit to particular environment conditions or to the limited number of airplanes that are seen during a single day.

TABLE I: Message Detection Confusion Matrix

| | Actual Message | Actual Silence |
|---|---|---|
| Pred. Message | 99.6 % | 0.5 % |
| Pred. Silence | 0.4 % | 99.5 % |

With this collected data, we train and test each of the networks separately. In Table I we report the confusion matrix of the message detection network and observe that the message detection network obtains a very high F1 score of 0.995. Furthermore, the network for offset calculation finds the starting point of the signal with an average deviation of $\pm 6.52$ samples, i.e., within $\pm 0.18\,\%$ of the total signal segment length.

Finally, our experiments show that on a test set of 50,000 messages, the bit decoding network is able to decode 93.1 % of the messages with no bit errors, 4.04 % with one bit error, 0.87 % with two bit errors and 1.97 % with more than two bit errors. Since the protocol allows for the correction of up to two wrong bits, our network can successfully decode 98.03 % of the messages that can be decoded by the baseline decoder.

These results show that the performance of our three staged pipeline is comparable to that of a classical decoder. Next, we study whether we can outperform the baseline by looking at messages that are not decoded by dump1090.

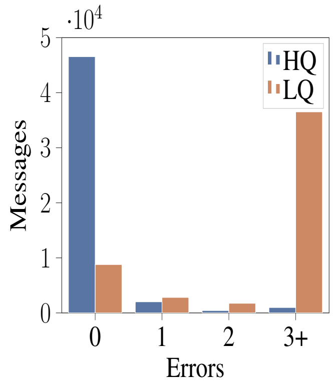
### B. Low quality samples

To evaluate whether our learning-based system can outperform the classical baseline decoder, we use the low quality samples that were recorded during the data collection process. As mentioned before, these messages were not detected and decoded by dump1090. Unlike a classical decoder, our system can be trained with low quality samples in order to make it more robust against interference and improve its decoding capabilities.

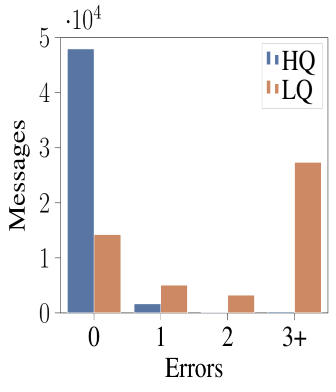
Therefore, we repeat the training procedure for all the three stages of the pipeline but now we include low quality (LQ) samples in the training set. We use a training set with a total of 600,000 messages of which 400,000 are high quality (HQ) messages and the remaining 200,000 are low quality, i.e., messages that are not decoded by our baseline, dump1090.

TABLE II: Message Detection Confusion Matrix

| | | Actual Mess. | | Actual Sil. | |
|---|---|---|---|---|---|
| | | HQ | LQ | HQ | LQ |
| HQT-Net | P. Mess. | 99.6 % | 99.5 % | 0.5 % | 69.5 % |
| | P. Sil. | 0.4 % | 0.5 % | 99.5 % | 30.5 % |
| LQT-Net | P. Mess. | 99.0 % | 98.5 % | 0.7 % | 4.5 % |
| | P. Sil. | 1.0 % | 1.5 % | 99.3 % | 95.5 % |



(a) HQT Network



(b) LQT Network

Fig. 1: Performance of the bit decoding network with HQ and LQ training and test sets.

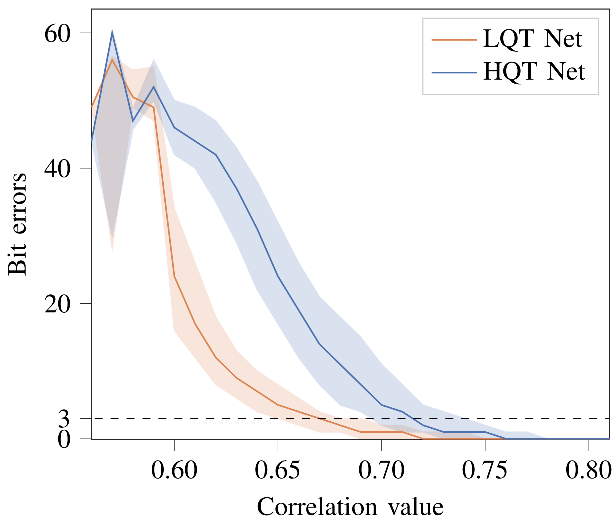


Fig. 2: Mean number of errors vs. message quality (correlation value)

To study how training on low quality data can improve the performance of our system, we compare each of the three networks in both configurations, trained only with high quality training data (HQT) and trained with high and low quality training data (LQT). In Table II we split the results between high quality and low quality samples and show the confusion matrix for each configuration. We observe that while both configurations obtain good results in HQ samples, the LQT configuration obtains a much higher F1 score of 0.970 on the low quality messages in comparison to the HQT network, which reaches only a F1 score of 0.467 due to a low true negative rate of only 30.5 %.

Similarly, the offset calculation networks perform very well on high quality messages: 6.5 samples mean deviation for the HQT network and 8.1 for the LQT network; but they differ significantly on low quality messages. Specifically, the HQT network finds the starting point of the message with a mean deviation of 114.3 samples for low quality messages whereas the LQT network suffers from a deviation of only 15.0 samples.

Finally, the decoding performance of our system is shown in Figures 1a and 1b. These figures show the number of messages with 0, 1, 2 and 3 or more bit errors for two test sets, one with 50,000 HQ messages and the other one with 50,000 LQ messages. We see in Figure 1a that the decoding network with HQT configuration can only decode 27.0 % of the low quality messages (less than 3 bit errors). Conversely, Figure 1b shows that using the LQT regime improves the decoding rate by almost 20 % and succeeds at decoding 45.3 % of the messages. Furthermore, training with LQ messages slightly improves the ability to correctly decode high quality messages, i.e., the LQT network fails to decode only 0.4 % while the HQT network fails for 2.0 % of the HQ messages.

### C. Deep learning vs. classical baseline

Despite the improvement achieved by using LQ samples during training, a significant amount of low quality messages cannot be recovered.

To understand which low quality messages can be decoded by our system, we plot in Figure 2, for both the HQT and the LQT networks, the number of decoding errors against the correlation values of the samples as obtained from the cross-correlation during the data collection stage. These results show that, for both networks, below a certain correlation value, the quality of the messages is not enough to be decoded reliably. After that point, the number of bit errors grows almost exponentially until it reaches randomness, that is, 56 bit errors on average.

It is easy to see that the number of errors in the LQT network is clearly shifted to the left in comparison to the HQT network. This implies that the LQT network can decode messages with lower correlation value, i.e., lower quality. In particular, on average it can correctly decode (less than 3 errors) messages down to a correlation value of 0.67.

To quantify the amount of messages that can be decoded by our LQT network and that cannot be decoded by the classical baseline, we analyze the distribution of messages as a function of the correlation value, i.e., signal quality. Figure 3 shows the distributions of HQ messages (decodable by the classical baseline) and LQ messages (non-decodable by the baseline). We normalize the area of both distributions and then calculate the area under the curve (AUC) of the HQ distribution and the AUC of the LQ distribution for correlation values larger than 0.67, i.e., the area of the LQ data to the right of the dashed line in Figure 3. This second AUC corresponds to the additional messages that our system in LQT configuration can decode with respect to dump1090. This calculation shows that assuming the same arrival rate for LQ and HQ messages, our system can decode 45.1 % more messages than the classical baseline.

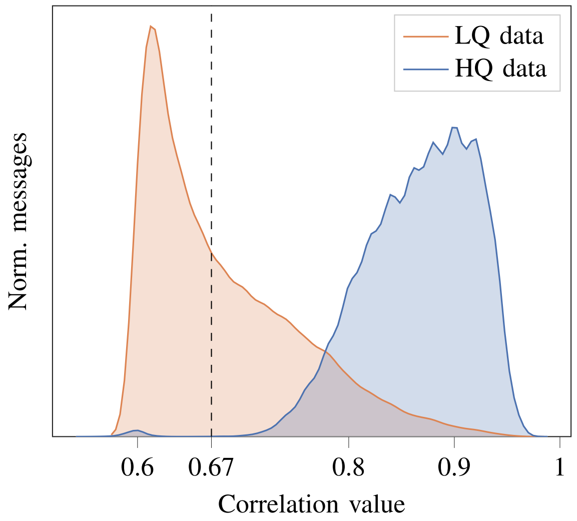


Fig. 3: Distribution of LQ and HQ messages with respect to message quality (correlation value)

## VII. Conclusion

In this work we have presented a deep learning-based decoding system for ADS-B aircraft messages. We break down the decoding task in three parts: message detection, offset calculation and bit decoding. Thus, our system consists of a three-stage pipeline with a specific neural network to solve each of the subtasks.

To collect the data needed to train this system, we use SDRs in several different locations. In this way, we have collected high quality data and low quality data that cannot be decoded locally by a classical decoder, dump1090, which we use as a baseline for evaluating the performance of our system.

Our results show that the performance of our deep learning-based decoder is on par with dump1090 when decoding local, high quality messages. Furthermore, we have shown that including low quality messages in the training data of our system is very effective and allows us to decode $45\,\%$ more messages than the classical baseline.

While our receiver has been trained to detect messages in one location, in the future it would be interesting to generalize the receiver to many reception conditions. An ADS-B receiver on an aircraft or mobile receivers for other protocols would largely benefit from such a generalized receiver.

The possibility of enhancing learning-based systems by using low quality data in the training phase permits improvements that can hardly be matched by classical decoders. Our system is tailored to decoding aircraft messages used for air traffic control. However, given the excellent performance of our system, we are positive that our approach would be successful if adapted for other communication protocols.

## References

[1] FAA, “Airspace,” https://www.faa.gov/nextgen/equipadsb/research/airspace/, accessed 2020-02-19.

[2] EASA, “Seasonal technical communication,” https://www.easa.europa.eu/sites/default/files/dfu/EASA_STC_NEWS_JUNE_2018.pdf, accessed 2020-02-19.

[3] O. Jowett, “Github - mutability/dump1090,” https://github.com/mutability/dump1090, accessed 2020-02-19.

[4] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, p. 436, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] T. Karras, T. Aila, S. Laine, and J. Lehtinen, “Progressive growing of gans for improved quality, stability, and variation,” *arXiv preprint arXiv:1710.10196*, 2017.

[7] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[8] S. Pascual, A. Bonafonte, and J. Serra, “SEGAN: Speech enhancement generative adversarial network,” *arXiv preprint arXiv:1703.09452*, 2017.

[9] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, “Deep Learning for Wireless Physical Layer: Opportunities and Challenges,” *CoRR*, vol. abs/1710.05312, 2017. [Online]. Available: http://arxiv.org/abs/1710.05312

[10] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be’ery, “Deep Learning Methods for Improved Decoding of Linear Codes,” *J. Sel. Topics Signal Processing*, vol. 12, no. 1, pp. 119–131, 2018. [Online]. Available: https://doi.org/10.1109/JSTSP.2017.2788405

[11] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, “On deep learning-based channel decoding,” in *51st Annual Conference on Information Sciences and Systems, CISS 2017, Baltimore, MD, USA, March 22-24, 2017*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CISS.2017.7926071

[12] H. Ye, G. Y. Li, and B. Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” *IEEE Wireless Commun. Letters*, vol. 7, no. 1, pp. 114–117, 2018. [Online]. Available: https://doi.org/10.1109/LWC.2017.2757490

[13] E. Habler and A. Shabtai, “Using LSTM encoder-decoder algorithm for detecting anomalous ADS-B messages,” *Computers & Security*, vol. 78, pp. 155–173, 2018. [Online]. Available: https://doi.org/10.1016/j.cose.2018.07.004

[14] M. Strohmeier, V. Lenders, and I. Martinovic, “Intrusion Detection for Airborne Communication Using PHY-Layer Information,” in *Detection of Intrusions and Malware, and Vulnerability Assessment - 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings*, 2015, pp. 67–77. [Online]. Available: https://doi.org/10.1007/978-3-319-20550-2_4

[15] ——, “Security of ADS-B: state of the art and beyond,” *CoRR*, vol. abs/1307.3664, 2013. [Online]. Available: http://arxiv.org/abs/1307.3664

[16] T. J. O’Shea, T. Erpek, and T. C. Clancy, “Deep learning based MIMO communications,” *CoRR*, vol. abs/1707.07980, 2017. [Online]. Available: http://arxiv.org/abs/1707.07980

[17] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “Deep learning based communication over the air,” *J. Sel. Topics Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018. [Online]. Available: https://doi.org/10.1109/JSTSP.2017.2784180

[18] J. L. Gertz, “Fundamentals of mode s parity coding,” Massachusetts Institute of Technology, Tech. Rep., 1984.

[19] M. Eichelberger, K. Luchsinger, S. Tanner, and R. Wattenhofer, “Indoor localization with aircraft signals,” in *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, SenSys 2017, Delft, Netherlands, November 06-08, 2017*. ACM, 2017, pp. 12:1–12:14. [Online]. Available: https://doi.org/10.1145/3131672.3131698

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.