Deep Joint Source-Channel Coding for Multi-Task Network

Mengyang Wang , Zhicong Zhang, Jiahui Li, *Member, IEEE*, Mengyao Ma, *Member, IEEE*, and Xiaopeng Fan , *Senior Member, IEEE*

Abstract—Multi-task learning (MTL) is an efficient way to improve the performance of related tasks by sharing knowledge. However, most existing MTL networks run on a single end and are not suitable for collaborative intelligence (CI) scenarios. In this work, we propose an MTL network with a deep joint source-channel coding (JSCC) framework, which allows operating under CI scenarios. We first propose a feature fusion based MTL network (FFMNet) for joint object detection and semantic segmentation. Compared with other MTL networks, FFMNet gets higher performance with fewer parameters. Then FFMNet is split into two parts, which run on a mobile device and an edge server respectively. The feature generated by the mobile device is transmitted through the wireless channel to the edge server. To reduce the transmission overhead of the intermediate feature, a deep JSCC network is designed. By combining two networks together, the whole model achieves $512\times$ compression for the intermediate feature and a performance loss within 2% on both tasks. At last, by training with noise, the FFMNet with JSCC is robust to various channel conditions and outperforms the separate source and channel coding scheme.

Index Terms—Collaborative intelligence, multi-task learning, deep joint source-channel coding.

I. INTRODUCTION

WITH the development of deep learning [1], the convolutional neural network (CNN) is playing an important role in computer vision tasks, like object detection [2] and semantic segmentation [3]. These tasks are usually studied as separate problems. However, preparing the model for each task individually (i.e., the so-called single-task learning, STL) is inefficient and storage-intensive. Therefore, multi-task learning (MTL) [4] emerges as the times require. In MTL, related tasks are handled using one model, where common features are obtained by sharing a backbone, and different branches are used to solve different tasks. Compared with STL, MTL is more efficient

Mengyang Wang and Zhicong Zhang are with the School of Computer Science, Harbin Institute of Technology, Harbin 150001, China, and also with the Wireless Technology Lab, Huawei, Shenzhen 518129, China (e-mail: wangmengyang4@huawei.com; zhangzhicong4@huawei.com).

Jiahui Li and Mengyao Ma are with the Wireless Technology Lab, Huawei, Shenzhen 518129, China (e-mail: lijiahui666@huawei.com; ma.mengyao@huawei.com).

Xiaopeng Fan is with the School of Computer Science, Harbin Institute of Technology, Harbin 150001, China (e-mail: fxp@hit.edu.cn).

and storage-saving. There are some excellent studies of joint object detection and semantic segmentation [5]–[7]. However, BlitzNet [5] and TripleNet [6] have multiple branches, skipping connections, which makes the backbone complex. DSPNet [7] has many layers and parameters, resulting in high model storage costs. None of these models are suitable for running on mobile devices.

To enable complex models running on mobile devices, a paradigm called collaborative intelligence (CI) [8] has emerged. In CI, a deep model is split between the mobile device and the edge server, which balances the computational load between them. The mobile device extracts and transmits intermediate features from the input signal, and the edge server receives and processes these features. To reduce the transmission overhead, some feature compression methods have been proposed in [9]–[12]. In [9] and [10], HEVC-intra and HEVC-inter are used to compress features, respectively. PNG and JPEG are used to compress features in [11] and [12]. But these models do not take into account transmission errors due to channel noise or interference.

In traditional communication systems, source coding and channel coding are two separate steps. Nowadays, joint source-channel coding (JSCC) is known to outperform the separate approach in practical applications [13]. Recently, the deep learning based joint source-channel coding method [14] has been proposed, which uses an auto-encoder (AE) to compress and transmit images or features over wireless channels. In [15] and [16], a CNN based AE is used to compress and transmit images. In [17]–[19], AE is used to compress the intermediate feature of a STL network. These deep models are trained with noise, so they are robust to channel interference.

To get an MTL network that is suitable for CI and robust to channel interference, in this letter, we propose an MTL network with deep JSCC for the intermediate feature. Firstly, an MTL network (FFMNet) is designed to handle object detection and semantic segmentation. Then FFMNet is split into mobile device part and edge server part. To reduce the transmission overhead, we propose a new AE based JSCC framework to compress and transmit the intermediate feature of FFMNet. Finally, the robust JSCC model for MTL is obtained through training the whole model with channel noise. To the best of the authors' knowledge, this is the first work to perform JSCC for MTL. Our main contributions are summarized as follows:

1) We propose a novel MTL network, FFMNet, for joint object detection and semantic segmentation. And FFMNet achieves better performance and has fewer parameters than other frameworks.
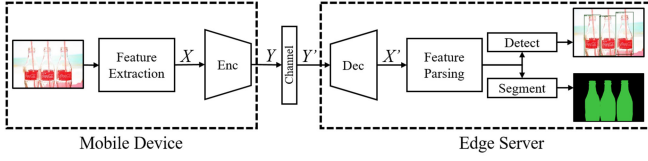
Fig. 1. The overall architecture of the proposed network.

2) A JSCC network is designed to compress and transmit the intermediate feature of FFMNet, which achieves $512\times$ compression with a performance loss under 2% on both tasks.

3) The FFMNet with JSCC model is trained to achieve robust transmission over noisy channel. Simulation results show that the performance of the proposed model is much better than that of the separate coding scheme.

The rest part of this paper is as follows. In Section II, we show the overall architecture, the details of the FFMNet and the JSCC model, and then introduce the loss functions and training strategy. The dataset and the performance of the FFMNet and the JSCC model are shown in Section III. Finally, the paper is concluded in Section IV.

II. METHODOLOGY

A. Overall Network Architecture

The overall architecture of the proposed framework is shown in Fig. 1. It contains a mobile device and an edge server. At the mobile device, the input image is first processed by a feature extraction module, which obtains deep intermediate feature X containing enough information for the considered tasks. Since X can be too large in size, transmitting it to the edge server requires significant communication resources. To reduce the communication overhead, an encoder compresses X into a compact feature Y, and then sends Y to the edge server through a noisy channel. At the edge server, noisy feature Y' is received and processed by a decoder to get the reconstruction X'. Finally, the edge server parses X' using the feature parsing module to obtain the features to be shared by object detection and semantic segmentation.

The wireless channel between the mobile device and the edge server is modeled by an additive white Gaussian noise (AWGN) model. Given an input feature $z \in \mathbb{R}^B$ and an output feature $z' \in \mathbb{R}^B$, the transfer function of AWGN channel is written as $z' = z + n$, with $n \sim \mathcal{N}(0, \sigma^2)$. The σ^2 is the noise variance, which denotes the channel condition. Since the function is differentiable, the network can be trained end-to-end. Besides, to meet the average transmit power constraint of $P = 1$, i.e. $\frac{1}{B}\sum_{i=1}^{B} z_i^2 = P$, a power normalization layer is put at the end of the encoder. We evaluate the performance of the object detection and semantic segmentation for different channel signal-to-noise ratios (SNRs) defined as $\frac{P}{\sigma^2}$.

B. Feature Fusion Based Multi-Task Network (FFMNet)

Nowadays, MTL is getting more attention, and there are many excellent frameworks. The BlitzNet [5] and TripleNet [6] perform object detection and semantic segmentation, and they exhibit high performance on both tasks at a high speed. In BlitzNet, ResNet50 [20] is used as the basic model, and the two
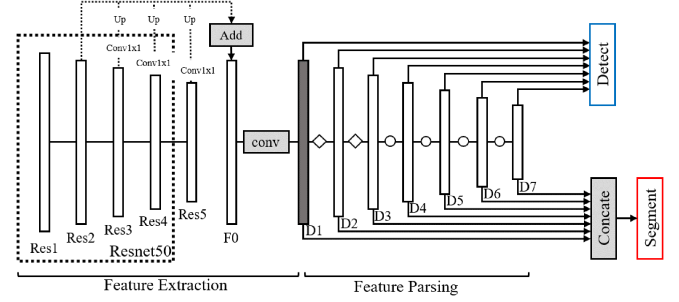

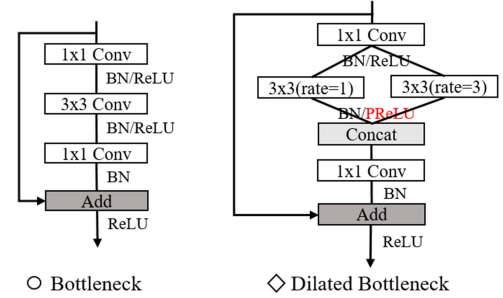Fig. 2. The architecture of FFMNet.

Fig. 3. The Bottleneck and Dilated Bottleneck used in feature parsing part.

branches perform their respective tasks by sharing a set containing multiscale features. There are many residual connections and 'ResSkip' blocks in the backbone to fuse the features of the front and back sides, which makes the backbone very complex. To get an MTL network that is lightweight and CI-suitable, we design a feature fusion based multi-task network FFMNet as shown in Fig. 2. In FFMNet, ResNet50 is also used as the basic model and multi-scale features are used by the two tasks. The difference with BlitzNet is the way of getting the multi-scale features set from the basic model. According to Fig. 2, FFMNet consists of three parts: feature extraction, feature parsing, and task branches.

In the feature extraction part, the Res_1 to Res_4 are generated by ResNet50 and Res_5 is an additional layer. Then the feature fusion technology similar to FSSD [21] is applied to fuse the Res_2 to Res_5 into one feature F_0. The dimension of F_0 is (64, 64, 512) and is represented by $(height, width, channels)$. We first use convolutional layers with 1×1 kernels to convert the channels of Res_2 to Res_5 into 512, then the bilinear upsampling method is used to convert the spatial dimensions of Res_2 to Res_5 into 64×64. After that, the resized features are added together as F_0. At last, the final feature D_1 is generated using a convolutional layer with 3×3 kernels.

In the feature parsing part, feature D_1 is parsed into a set of multi-scale features to perform the two tasks. The scales from feature D_1 to D_7 are (64, 64, 512), (32, 32, 512),..., (1, 1, 512), respectively. The reason for using multi-scale features is that different scale features have different reception fields, which helps to identify objects of different scales [22]. As shown in Fig. 3, the 'Bottleneck' of ResNet and 'Dilated Bottleneck' are used to generate features with different scales, and the 'Dilated Bottleneck' is formed based on 'Bottleneck' and dilated
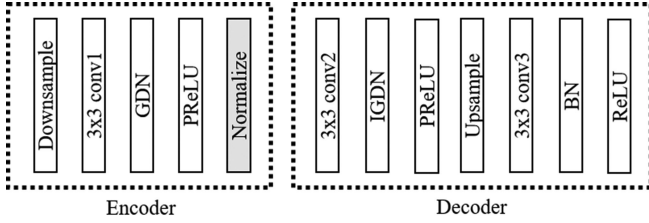
Fig. 4. The proposed JSCC encoder and decoder.

TABLE I
COMPARISON BETWEEN MTL NETWORKS

Network	mAP(%)	mIoU(%)	Param
FFMNet	40.8	**44.6**	**63.2M**
BlitzNet	40.1	44.1	87.8M
TripleNet	**41.4**	44.0	137.3M

convolutional layer. Since the sizes of D_1 and D_2 are much bigger than others, the 'Dilated Bottleneck' with larger receptive field is employed. D_2 and D_3 are generated by the 'Dilated Bottleneck'. The remaining small-scale features are generated by 'Bottleneck'.

After getting the feature set, object detection head and semantic segmentation head share the features to perform their own tasks, and these heads are the same as those used in BlitzNet. In the detection branch, there are two convolutional layers with 3×3 kernels for classification and location regression respectively for each size of features in the set. Then the non-maximum suppression (NMS) is employed as the post-processing method to eliminate redundant detection results. On the other side, the segmentation branch first uses upsampling method and 1×1 convolutional layer to resize the features into the same size (64, 64, 64). Then the rescaled features are concatenated together in channel dimension. At last, a convolutional layer with 3×3 kernels is to predict the classes of each pixel of the image.

C. JSCC of Intermediate Feature

With the idea of CI and the structure of FFMNet introduced before, we split FFMNet into two parts and apply a JSCC network on the intermediate feature. We choose D_1 as the split point and propose an AE based JSCC network. The structure of the JSCC network is shown in Fig. 4.

Inspired by [18], an asymmetric structure is designed to reduce computation on the device, so there are fewer layers in the encoder than in the decoder. In the encoder, the downsampling method is proposed to reduce the spatial size of the feature, and the number of feature channels is reduced by a convolutional layer with 3×3 kernels. Then, the generalized divisive normalization (GDN) [23] layer follows the convolutional layer. At the end of the encoder, we place a power normalization layer. In the decoder, two convolutional layers with 3×3 kernels recover the channels gradually, and an upsampling layer between them is used to recover the spatial size. The inverse generalized divisive normalization (IGDN) layer performs the inverse operation of the GDN layer.

Compare to the AE in [18], the proposed JSCC network has some advantages: First, the downsampling layer can downsample the spatial size with stride 4, which can not be done with 3×3 convolutional layers. Second, in the decoder, we use two convolutional layers to recover the channels gradually, which helps to reconstruct the feature. Third, the last activation function is ReLU, which makes the reconstructed feature in the same value range as D_1.

By combining this network with the FFMNet, we finally get the JSCC architecture for MTL.

D. Training Strategy

When training FFMNet, the loss function $\mathcal{L}_{MTL}$ has two parts: the loss function of object detection $\mathcal{L}_{det}$ and semantic segmentation $\mathcal{L}_{seg}$. For detection, we use the same loss function in SSD [22], which includes a classification loss $\mathcal{L}_{cls}$ and a regression loss of locating the bounding boxes $\mathcal{L}_{bbox}$. So the loss function of detection is

$$\mathcal{L}_{det} = \mathcal{L}_{cls} + \mathcal{L}_{bbox}. \tag{1}$$

For segmentation, the loss is the cross-entropy between predicted and target class distribution. We add the loss functions of two branches as the final loss function:

$$\mathcal{L}_{MTL} = \mathcal{L}_{det} + \mathcal{L}_{seg}. \tag{2}$$

To fully train the network, we propose a three-step training strategy: Firstly $\mathcal{L}_{MTL}$ is employed to train the FFMNet. Then we attach the JSCC network at the split point, the sum of the L1 loss between X and X' and the $\mathcal{L}_{MTL}$ is the loss function to train the JSCC part, and other parameters are fixed. Finally, we use $\mathcal{L}_{MTL}$ to train the parameters of the whole network end-to-end.

To make the whole system robust to the channel noise, the network will be trained at different SNRs by changing the σ^2.

III. EXPERIMENTS

A. Dataset

In the experiments, we use part of the Open Images Dataset [24] as training and testing data. The training set contains 178847 images and testing set contains 9903 images. All the images are labeled with the annotations of object detection and semantic segmentation, and there are 117 object categories in the annotation.

B. Performance for FFMNet

At the first step of the training strategy, FFMNet is optimized by the Adam algorithm with a mini-batch size of 32 images. The iteration number is 500 K, and the initial learning rate is set to 10^{-4} and decreased twice by factor 10 at iteration 250 K and 450 K.

According to the training settings, FFMNet is well trained. In Table I, we compare the performance on two tasks and the number of parameters for the entire network of FFMNet, BlitzNet and TripleNet. The metric for evaluating detection performance is the mean average precision (mAP) and the quality of predicted segmentation masks is measured with mean intersection over union (mIoU). It is shown that FFMNet performs better than BlitzNet while has fewer parameters. Compared with TripleNet, FFMNet performs slightly worse on detection but better on segmentation. Besides, the number of parameters is reduced by

TABLE II
COMPARISON BETWEEN MTL AND STL

Network	Det	Seg	mAP(%)	mIoU(%)
FFMNet	✓	✓	**40.8**	**44.6**
FFMNet	✓		40.3	-
FFMNet		✓	-	40.4

TABLE III
TASK PERFORMANCE VS. COMPRESSION RATIO

Feature Size	Compression Ratio	mAP(%)	mIoU(%)
(H,W,C)	-	40.8	44.6
(H/2,W/2,C/32)	$128\times$	40.2 (-0.6)	43.6 (-1.0)
(H/2,W/2,C/64)	$256\times$	39.7 (-1.1)	43.1 (-1.5)
(H/4,W/4,C/32)	$512\times$	39.4 (-1.4)	43.1 (-1.5)
(H/4,W/4,C/64)	$1024\times$	38.8 (-2.0)	42.4 (-2.2)

more than 50%, making FFMNet more suitable for CI scenarios than the baseline methods.

To verify whether MTL outperforms STL, we train two STL networks respectively, and the results are shown in Table II. As can be seen from Table II, joint detection and segmentation improves detection and segmentation performance by 0.5% and 4.2%, respectively, indicating that the two tasks are mutually beneficial in FFMNet.

C. Performance for JSCC Model

The proposed JSCC network is trained by the strategy described in Section II-D, on the basis of FFMNet.

In the second step of the training strategy, we train the encoder and decoder for 100 K iterations, the learning rate is set to 10^{-4} and decreased by 10 at iteration 50 K. In the third step, the whole network is trained end-to-end for 200 K iterations, the learning rate is set to 10^{-5} and decreased by 10 at iteration 100 K and 150 K.

To explore the compression capability of the JSCC model, a group of JSCC models with different compression ratios for the intermediate feature of FFMNet are trained under the noiseless condition. The performance of different JSCC models is shown in Table III, where the original feature dimension is (H, W, C). With performance degradation threshold set to 2% on both tasks, the JSCC model achieves up to $512\times$ compression of the intermediate feature.

In addition to the compression capability, robustness also needs to be considered in practice. To find the best training SNR (i.e., SNR_{train}) for JSCC, multiple networks are trained with channel noise under different SNRs, i.e., $\text{SNR}_{train} = 0$, 5, and 10 dB, where the compression ratio is fixed to $512\times$.

We plot the mAP of detection and the mIoU of segmentation under different test channel SNRs in Fig. 5. In the figure, there are two baselines. The first baseline is the performance of the FFMNet without compression. The second baseline is the performance of the JSCC model with $512\times$ compression, which is trained and tested without channel noise. Besides, the performance of the proposed deep JSCC for the intermediate feature is compared with separate schemes that use JPEG for feature compression followed by practical channel coding and modulation.
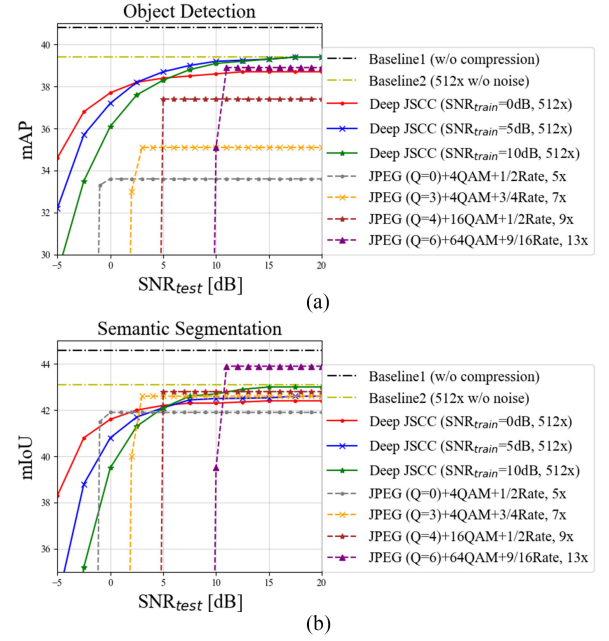

Fig. 5. Performance of detection and segmentation with respect to testing SNR (i.e., SNR_{test}).

For separate scheme, we first perform 8-bit uniform quantization on feature. Then, JPEG with different quality parameters is used to compress the quantized feature. At last, channel coding and modulation are employed. We take four combinations of source and channel coding to get the similar performance of the proposed JSCC method. It is obvious that the JSCC method outperforms the separate scheme in Fig. 5. On one hand, with similar performance, the maximum compression ratio of the separate scheme is $13\times$ while JSCC is $512\times$. On the other hand, the proposed JSCC model does not suffer the 'cliff effect' in separate method.

For JSCC, with the decrease of SNR_{train}, although the two task branches become more robust to the noise at low SNR values, the performance of both tasks degrades a little at high SNR. To get the trade-off between performance and robustness, the model trained with $\text{SNR}_{train} = 5$ dB behaves well, so we select it as the final JSCC model for MTL. The final model is CI-suitable and robust to channel inference.

IV. CONCLUSION

In this letter, we study the MTL network which is CI-suitable, lightweight, and robust to channel interference. We first propose an MTL network named FFMNet, which performs object detection and semantic segmentation in the meantime. Results show that FFMNet gets higher performance and has fewer parameters than the baseline methods. Then we split the FFMNet into two parts and propose a JSCC scheme for efficient feature compression and robust feature transmission over the AWGN channel. The JSCC scheme achieves $512\times$ compression for the intermediate feature. Besides, the model trained with $\text{SNR}_{train} = 5$ dB exhibits robustness over a wide range of SNR_{test} and outperforms the separate method a lot.

REFERENCES

[1] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep Learning.* Cambridge, MA, USA: MIT Press, 2016, vol. 1, no. 2.

[2] I. Park and S. Kim, "Performance indicator survey for object detection," in *Proc. 20th Int. Conf. Control, Automat. Syst.*, 2020, pp. 284–288.

[3] F. Lateef and Y. Ruichek, "Survey on semantic segmentation using deep learning techniques," *Neurocomputing*, vol. 338, pp. 321–348, 2019.

[4] Y. Zhang and Q. Yang, "An overview of multi-task learning," *Nat. Sci. Rev.*, vol. 5, no. 1, pp. 30–43, 2018.

[5] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "Blitznet: A real-time deep network for scene understanding," in *Proc. IEEE Int. Conf. Comput. Vision*, 2017, pp. 4154–4162.

[6] J. Cao, Y. Pang, and X. Li, "Triply supervised decoder networks for joint detection and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 7392–7401.

[7] L. Chen, Z. Yang, J. Ma, and Z. Luo, "Driving scene perception network: Real-time joint detection, depth estimation and semantic segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vision*, 2018, pp. 1283–1291.

[8] Y. Kang *et al.*, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," *ACM SIGARCH Comput. Architecture News*, vol. 45, no. 1, pp. 615–629, 2017.

[9] H. Choi and I. V. Bajić, "Deep feature compression for collaborative object detection," in *Proc. 25th IEEE Int. Conf. Image Process.*, 2018, pp. 3743–3747.

[10] H. Choi and I. V. Bajić, "Near-lossless deep feature compression for collaborative intelligence," in *Proc. IEEE 20th Int. Workshop Multimedia Signal Process.*, 2018, pp. 1–6.

[11] S. R. Alvar and I. V. Bajić, "Multi-task learning with compressible features for collaborative intelligence," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 1705–1709.

[12] S. R. Alvar and I. V. Bajić, "Bit allocation for multi-task collaborative intelligence," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 4342–4346.

[13] F. Zhai, Y. Eisenberg, and A. K. Katsaggelos, "Joint source-channel coding for video communications," in *Handbook of Image and Video Processing.* New York, NY, USA: Elsevier-Academic, 2005, pp. 1065–1082.

[14] L. Rongwei, W. Lenan, and G. Dongliang, "Joint source/channel coding modulation based on BP neural networks," in *Proc. Int. Conf. Neural Netw. Signal Process.*, 2003, pp. 156–159.

[15] E. Bourtsoulatze, D. B. Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 3, pp. 567–579, Sep. 2019.

[16] D. Burth Kurka and D. Gündüz, "Joint source-channel coding of images with (not very) deep learning," in *Proc. Int. Zurich Seminar Inf. Commun.*, 2020, pp. 90–94.

[17] J. Shao and J. Zhang, "BottleNet: An end-to-end approach for feature compression in device-edge co-inference systems," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2020, pp. 1–6.

[18] M. Jankowski, D. Gunduz, and K. Mikolajczyk, "Joint device-edge inference over wireless links with pruning," in *Proc. IEEE 21st Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, 2020, pp. 1–5.

[19] M. Jankowski, D. Gündüz, and K. Mikolajczyk, "Deep joint source-channel coding for wireless image retrieval," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 5070–5074.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.

[21] Z. Li and F. Zhou, "FSSD: Feature fusion single shot multibox detector," 2017, *arXiv:1712.00960*.

[22] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vision*, 2016, pp. 21–37.

[23] J. Ballé, V. Laparra, and E. P. Simoncelli, "Density modeling of images using a generalized normalization transformation," 2015, *arXiv:1511.06281*.

[24] A. Kuznetsova *et al.*, "The open images dataset V4," *Int. J. Comput. Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.