DeepJSCC-l++: Robust and Bandwidth-Adaptive Wireless Image Transmission

Chenghong Bian, Yulin Shao, Deniz Gunduz

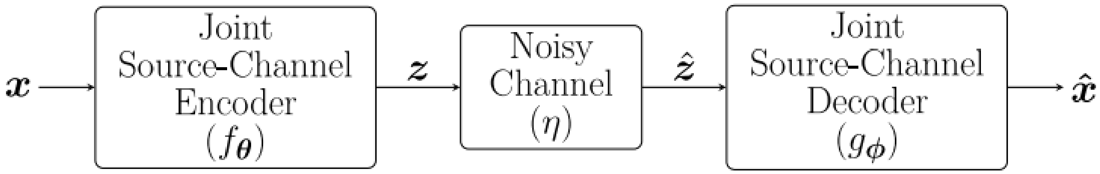
Imperial College London

DeepJSCC-l++

- **Outline**
  - Introduction & Motivation
  - System Model
  - Proposed Method
  - Experimental Results
  - Q & A

Introduction & Motivation

- DeepJSCC revisit
  - An encoder $f_\theta(\cdot)$ and decoder $g_\phi(\cdot)$ parameterized by neural networks.
  - $x \in \mathbb{R}^{C\times H\times W}, z \in \mathbb{C}^k$, with $\rho = \frac{k}{N}, \mathrm{N} = \mathrm{C} \times H \times W$.
  - Jointly optimize $\{\theta, \phi\}$ to minimize reconstruction distortion, e.g., $||x - \hat{x}||_2^2$.



  - Key merit:
    - avoiding cliff and leveling effect
    - Achieving better reconstruction performance.

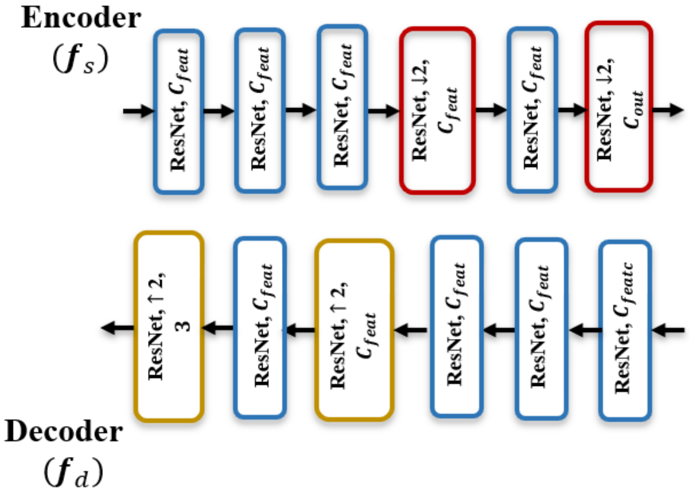
  - Performance metrics:
    - $PSNR = 10\log_{10}\frac{255^2}{\frac{1}{N}||x-\hat{x}||_F^2}$

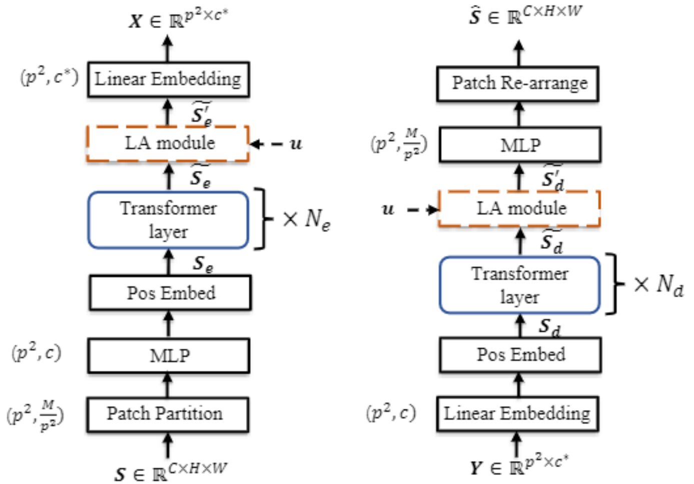
[1] E. Bourtsoulatze, D. B. Kurka, and D. Gunduz, "Deep joint sourcechannel coding for wireless image transmission," IEEE Trans. Cognitive Commun. Netw., vol. 5, no. 3, pp. 567–579, 2019.

Introduction & Motivation

- Motivation for DeepJSCC-l++
  - $\{\theta, \phi\}$ parameterized by CNNs/ViT, occupy 100+ Mb.
  - $\{\theta, \phi\}$ are optimized for a specific $\{SNR, \rho\}$.
  - 4 different SNR points and 4 bandwidth ratios → 16 models → 1.6 Gb space. **-- Too large for a mobile device!**

  - **Is it possible to have a single $\{\theta, \phi\}$ pair for all $\{SNR, \rho\}$?**



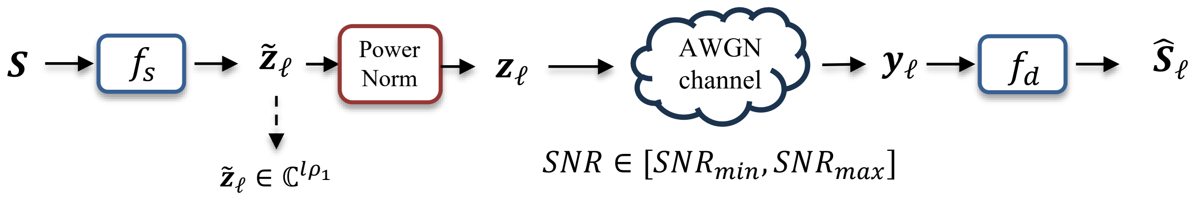


Possible neural network architectures to parameterize $\{\theta, \phi\}$:

Left: CNN,
Right: ViT.

System Model

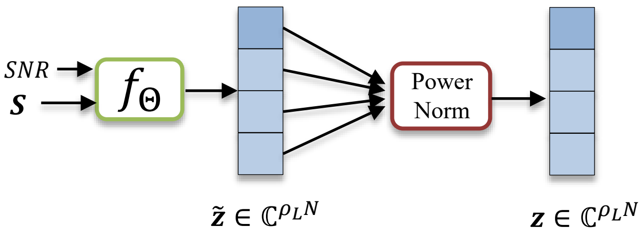
- System Model
  - Communication over an AWGN channel,
    - $\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{n}$; $\boldsymbol{z} \in \mathbb{C}^k, n_i \sim \mathcal{CN}(0, \sigma^2)$.
  - Assumptions of packets:
    - Possible supported bandwidth ratios $[\rho_1, \ldots, \rho_L]$ with $\rho_l = l\rho_1$. (discrete)
    - Channel qualities: $SNR \in [SNR_{min}, SNR_{max}]$ (continuous).
  - Channel model shown below:



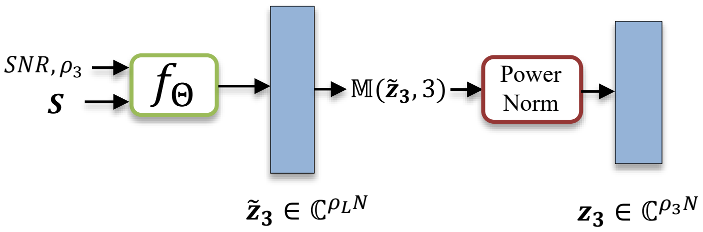
Proposed Method

- Solutions
  - Solution A: Successive Refinement
  - SNR-adaptive solution is given in [2], first focus on the varying bandwidth setting.

  - Shown in figure (a):
    - $\tilde{\boldsymbol{z}} = \boldsymbol{f}_{\boldsymbol{\Theta}}(\boldsymbol{S}, \boldsymbol{SNR})$
    - Partition $\tilde{\boldsymbol{z}}$ to L blocks, denote it as $\tilde{\boldsymbol{z}}[\boldsymbol{l}], \boldsymbol{l} \in [\mathbf{1}, \boldsymbol{L}]$.
    - ***Block-wise*** Power normalize: $\boldsymbol{z}[\boldsymbol{l}] = \frac{\tilde{\boldsymbol{z}}[\boldsymbol{l}]}{||\tilde{\boldsymbol{z}}[\boldsymbol{l}]||_2}$.
    - The first $k \in [1, L]$ blocks should be able to reconstruct the image, $\hat{S}_l$ to some level.
    - Loss function:
      - $\mathcal{L} = \sum_{l=1}^{L} ||S - \hat{S}_l||_F^2$



(a) Successive refinement



(b) Bandwidth adaptive

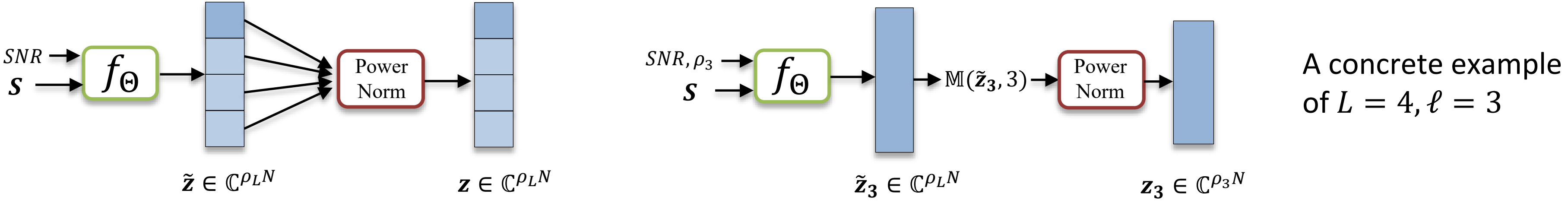
A concrete example of $L = 4, \ell = 3$

[2] J. Xu, etc, "Wireless image transmission using deep source channel coding with attention modules," IEEE Trans. Circuits Syst. Video Technol. 2021

Proposed Method

- Solution B: Bandwidth Adaptive; in figure (b)
  - Obtain target bandwidth, ℓ, from the user.
  - $\tilde{\boldsymbol{z}}_\ell = \boldsymbol{f}_{\boldsymbol{\Theta}}(\boldsymbol{S}, \boldsymbol{SNR}, \ell)$
  - Masking instead of partitioning:
    - $\boldsymbol{z}_\ell = \boldsymbol{M}(\tilde{\boldsymbol{z}}_\ell, \ell)$
  - Train the model as:
    - $\mathcal{L} = \mathbb{E}_{\ell \sim \mathcal{U}(1,L)} ||\boldsymbol{S} - \widehat{\boldsymbol{S}}_\ell||_F^2$

– Compare with Solution A:
  - Additional information from the user is required.
  - The reconstruction can be performed when all the $\boldsymbol{z}_\ell$ is received.
  - Since different users have different requirements, ℓ, in general, it only fits the point-to-point channel, instead of broadcast channel (where solution A can be easily applied).



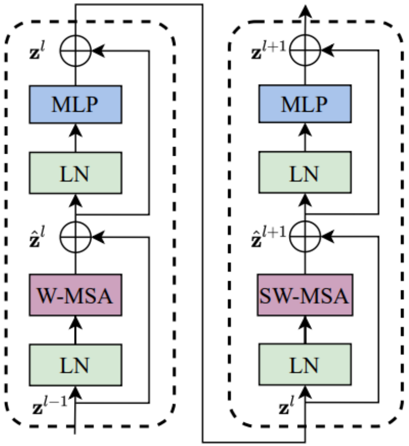
(a) Successive refinement

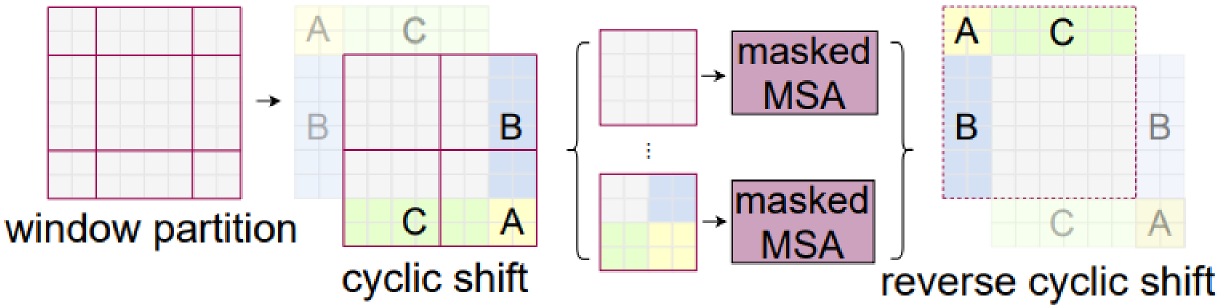
(b) Bandwidth adaptive

Proposed Method

- A good backbone – Swin Transformer
  - A good neural network backbone makes a difference – an improvement from ViT to CNN is observed [3]
  - Make it even better by adopting the state-of-the-art Swin transformer.

  - A brief introduction below:
    - The key idea for Swin is to perform self attention within a (small, e.g., 7x7) window instead of the whole image.
    - Shifting window operations enable communication between different windows.
    - ***Swin Transformer Block*** (Left) Illustration of SW-MSA module (Right).

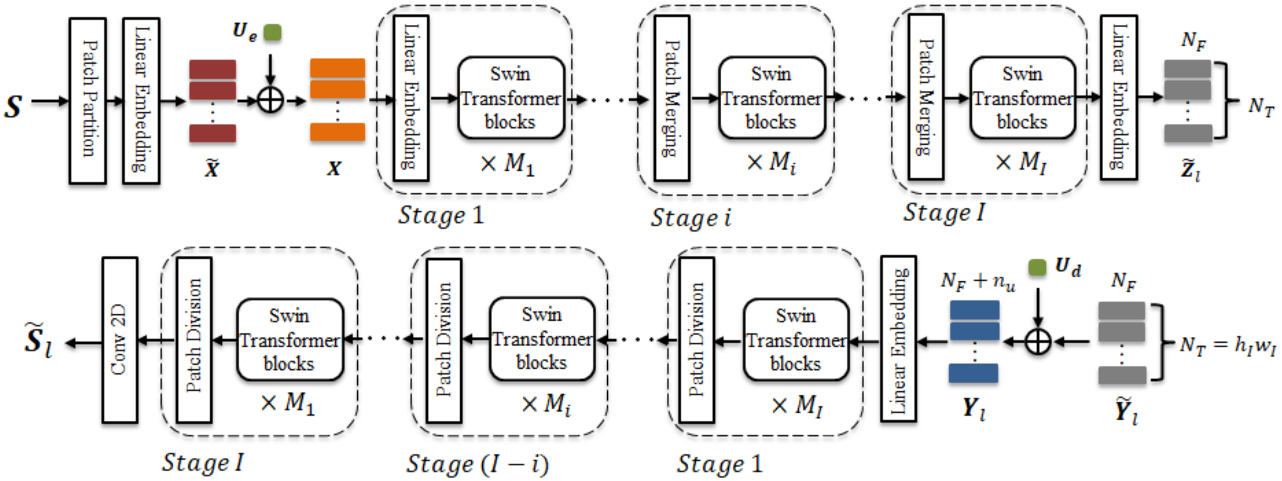




[3] H. Wu, etc, "Vision transformer for adaptive image transmission over MIMO channels," in IEEE International Conference on Communications (ICC), 2023.
[4] Z. Liu, etc,, "Swin transformer: Hierarchical vision transformer using shifted windows," in ICCV, October 2021

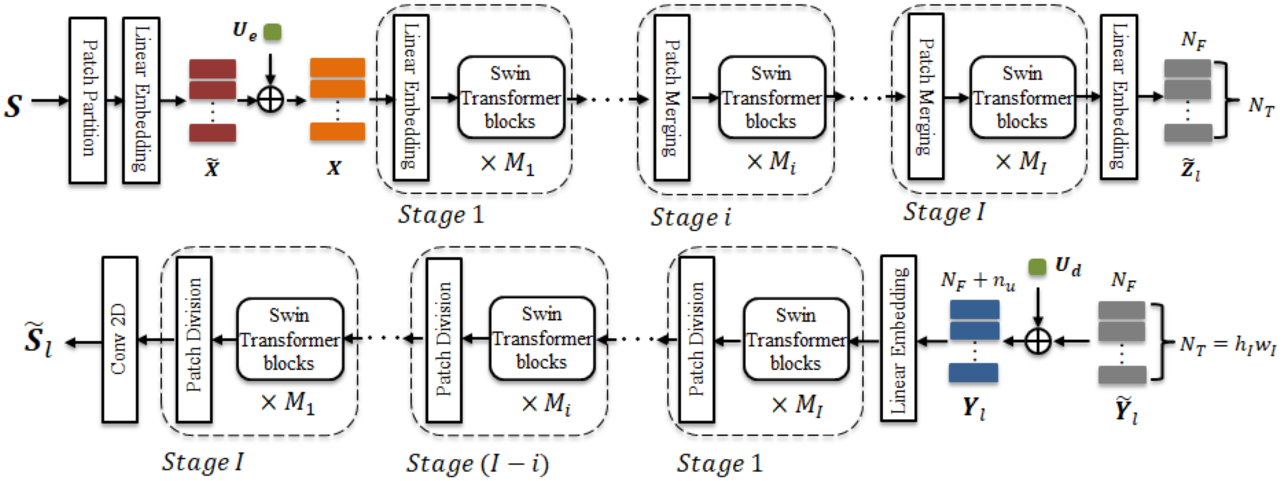
Proposed Method

- The overall framework
  - Explaining details of the framework:
    - Patch partitioning: $S\ (C,H,W) \rightarrow X'\left(Cp^2, h=\frac{H}{p}, w=\frac{W}{p}\right)$ -- dividing the image evenly.
    - Patch Merging: $X(d,h',w') \rightarrow X'\left(d', \frac{h\prime}{2}, \frac{w\prime}{2}\right)$ -- can be understood as stacking the patches and transform.
    - Patch Division: $X(d,h',w') \rightarrow X'(d',2h',2w')$ -- can be achieved via 2d transpose convolution.



Proposed Method

- The overall framework
  - Side information generation:
    - $\mathrm{U} = MLP([\rho_\ell, SNR])$, Note, if successive refinement, we use only SNR values.
    - Concatenate U to each token of $\tilde{X}$ $(\mathrm{d}, \mathrm{h}, \mathrm{w})$ to generate X $(d + n_{emb},\ h,\ w)$.
  - We show by experiments, we can achieve both SNR and bandwidth adaptive objective by feeding $MLP([\rho_\ell, SNR])$ to the framework.
  - ***Contrary to [2], we don't need to add the SA block to achieve SNR-adaptive! The Swin transformer performs it automatically.***



<u>Varying features vs varying tokens:</u>

- The output of the encoder is a matrix with $N_T \times N_F$.
- For different ρ, we can either transmit less amount of N_t or less amount of N_f. Simulations show different strategies yield similar reconstruction performance.

Proposed Method

- Training methodology

  - Naïve training strategy for Solution A (successive refinement):
    - Recall: total bandwidth budget, ρ_L, is fixed, number of layers, L, is fixed, the first l layers should recover the original image to some level…
    - $\mathcal{L} = \sum_{l=1}^{L} ||\boldsymbol{S} - \hat{\boldsymbol{S}}_l||_F^2$ if noise power is fixed, if not:
    - $\mathcal{L} = \mathbb{E}_{\gamma \sim \mathcal{U}(\gamma_1, \gamma_2)} \sum_{l=1}^{L} ||\boldsymbol{S} - \hat{\boldsymbol{S}}_{l,\gamma}||_F^2$, where γ_1, γ_2 denote the min and max SNR value.

  - Naïve training strategy for Solution B (bandwidth adaptive):
    - Recall: user specific bandwidth ratio ρ_ℓ is known to the transmitter, ℓ is a random variable.
    - $\mathcal{L} = \mathbb{E}_{\ell \sim \mathcal{U}(1,L)} ||\boldsymbol{S} - \hat{\boldsymbol{S}}_\ell||_F^2$ if noise power is fixed, if not:
    - $\mathcal{L} = \mathbb{E}_{\gamma \sim \mathcal{U}(\gamma_1, \gamma_2), \ell \sim \mathcal{U}(1,L)} ||\boldsymbol{S} - \hat{\boldsymbol{S}}_{\ell,\gamma}||_F^2$
    - This is achieved by randomly sample γ, ℓ from the uniform distributions and then calculate the end-to-end loss.

    - All the equations assume a same weight $w_\ell = 1$ for all the bandwidth ratios.

Proposed Method

- Dynamic weight assignment (DWA)
  - Problem of Naïve training for Solution A&B:
    - Having $w_\ell = 1$ for all bandwidth ratios results in:
      - the model only focus on optimizing the lower ρ_l with much larger loss $\mathcal{L}_\ell$.
      - $PSNR_\ell$ of the higher bandwidth ratios are much worse than $PSNR_l^*$ of the separately trained models.
    - This will be verified in the experiments.

  - **Solution:**
    - Consider unequal w_ℓ to different ρ_ℓ's.
    - However, it is hard to figure out a good configuration of $\boldsymbol{w}$ to achieve a good overall performance.

  - Instead of pre-assigned values, we have:
    - **Dynamic weight assignment: assigning larger w_l^t to $\mathcal{L}_l^t$ with larger ρ_l dynamically during the training epochs t.**

Proposed Method

- Dynamic weight assignment (DWA)
  - The weights w_l^t are updated according to the reconstruction performance evaluated over the validation dataset:
    - The separately trained models with $[\rho_1, \ldots, \rho_L]$ yields $[PSNR_1^*, \ldots, PSNR_L^*]$
    - Use it as a criterion to guide the training process of DeepJSCC-l++
  - we first determine the gap from the optimal results, then calculate the weight for each bandwidth ratio.

$$\Delta_l^t = \mathrm{PSNR}_l^* - \mathrm{PNSR}_l^t,$$
$$w_l^t = \mathrm{clip}(2^{\alpha(\Delta_l^t - \beta)} - 1,\ 0, \Gamma), \tag{5}$$

  - Explanation of the Parameters
    - α: how sensitive the weight w.r.t to the gap.
    - β: allowable gap from the optimal PSNR value.
    - Γ: designed for stable training, should not be too large.
    - The function in (5) is continuous from $[0, \Gamma]$.

Proposed Method

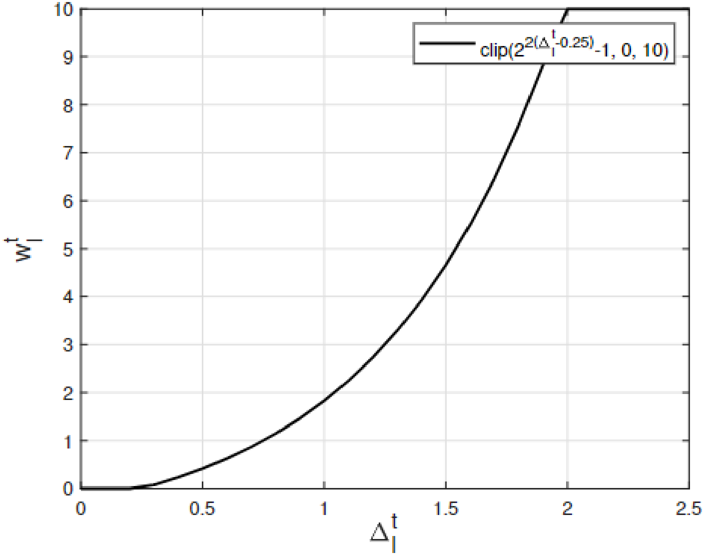
- A summary of the DeepJSCC-l++ framework

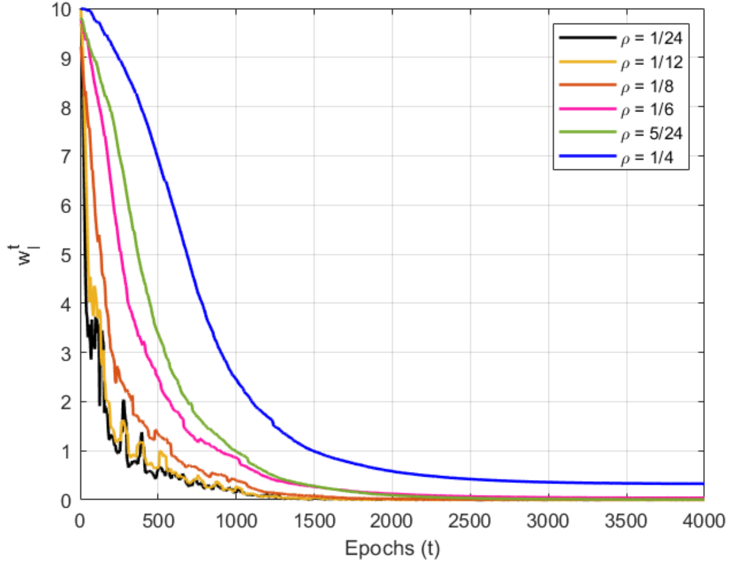
Algorithm 1 Overall Training Process for DeepJSCC-l++ Model with DWA.

Initialize $w_l^1 = 1, \forall l \in [L]$
for $t = 1, \ldots, T$ **do**
  Training Phase:
  for each batch **do**
    Sample $l \in [L], \text{SNR} \in [\text{SNR}_{min}, \text{SNR}_{max}]$
    Encoder: $\boldsymbol{z}_l = f_\Theta(\boldsymbol{S}, \text{SNR}, l)$
    Decoder: $\widetilde{\boldsymbol{S}}_l = g_\Psi(\boldsymbol{y}, \text{SNR}, l)$
    Weighted Loss: $\mathcal{L}_l^t = w_l^t ||\boldsymbol{S} - \widetilde{\boldsymbol{S}}_l||_2^2$.
    Optimize $\{\Theta, \Psi\}$ using $\mathcal{L}_l^t$.
  Validation Phase:
  for $l \in [L]$ **do**
    Calculate $\text{PSNR}_l^t, \Delta_l^t$ over validation set.
    Update w_l^t according to (5).

Simulation Results

- Analysis of DWA
  - Given parameters: $(\alpha, \beta, \Gamma) = (2, 0.25, 10)$.
  - Left figure → the w_l^t v.s. Δ_l^t.
  - Right figure → We train a bandwidth adaptive DeepJSCC-l++ (solution B) with $L = 6$ and 4000 epochs.
    - Larger bandwidth ratio needs larger weight.
    - The gap for $\rho_L = 1/4$ is larger than $\beta = 0.25$ dB even when the training ends.



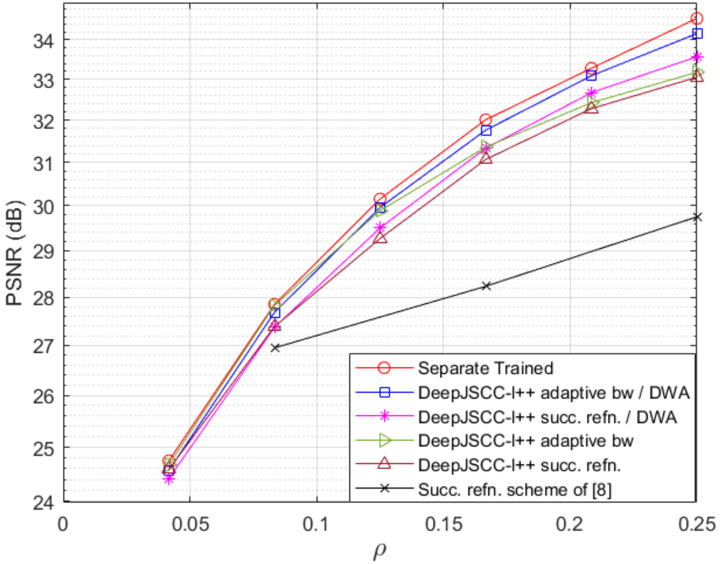


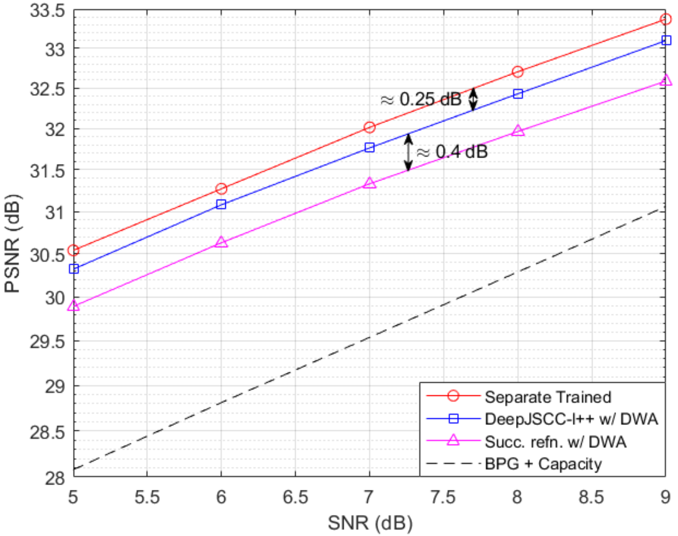
Simulation Results

- Bandwidth & SNR adaptive
  - Experiment setup
    - CIFAR-10 dataset, 32 x 32 resolution
    - Swin encoder/decoder employ I = 2 stages with the numbers of Swin transformer blocks in each stage is set to $M_1 = 4, M_2 = 2$, the number of features c is set to 256, the window size to w = 8,.
    - The dimension of the embedding is set to 2.
    - Train for 4000 epochs, varying learning rate initialized at 10−4 , which is reduced by a factor of 0.95 if the validation loss does not drop for 20 epochs.
    - DeepJSCC-l++ models for Solution A & B are trained under the conditions:
      - $SNR \in [4, 10]$ dB
      - $\rho_l \in \left[\frac{1}{24}, \frac{1}{12}, \frac{1}{8}, \frac{1}{6}, \frac{5}{24}, \frac{1}{4}\right]$
  - Benchmark:
    - 1. Separately trained models -- upper bound for the proposed DeepJSCC-l++
    - 2. BPG compression algorithm delivered at the (AWGN) channel capacity.

Simulation Results

- Bandwidth & SNR adaptive
  - Left: PSNR under (fixed) $SNR = 7$ dB, with varying bandwidth ratios.
    - DWA improves the reconstruction performance a lot at larger ρ_l yet only sacrifice very little PSNR at small ρ_l.
    - Solution B can outperform Solution A as its encoding phase is more flexible (known the target ρ_l)
  - Right: PSNR under (fixed) $\rho_l = \frac{1}{6}$, with varying channel qualities.
    - The proposed schemes can significantly outperform the digtial baseline while avoiding cliff-effect.
    - Adapting to SNR does not sacrifice the performance at all, the gap between the optimal is from the bandwidth adaptive part.





Simulation Results

- Supplementary materials
  - Left → Compare the Swin transformer with ViT.
    - Configuration: SNR = 7 dB, separately trained.
  - Right → varying patches/tokens v.s. varying features.
    - Swin Transformer, with architecture shown in Slide 10.

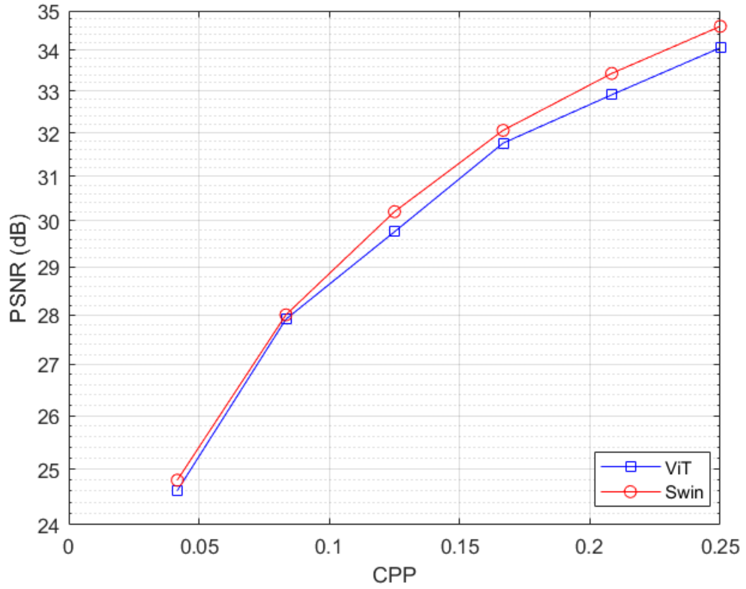


Table I: Evaluation for the *varying patches* and *varying features* DeepJSCC-l++ schemes at SNR $= 7\ dB$ in terms of PSNR (dB).

ρ	**1/16**	**1/8**	**3/16**	**1/4**
varying patches	26.12	30.01	32.53	34.32
varying features	26.14	30.01	32.53	34.31
separate training	26.36	30.23	32.70	34.55

Q&A

Thanks for Listening!

Any questions are welcomed! ☺