

A Theoretical Derivation

A.1 Proof of Equation 3 (Optimal Framework for Activation Sparsity-Aware Pruning)

We provide the detailed derivation of **DuoGPT**'s optimal calibration framework (Equation 3) proposed in Section 4.2.

Proof. The Lagrangian in Equation 2 is first expanded:

$$\begin{aligned}\mathcal{L} &= \text{Tr}\left((\Delta\mathbf{w}\hat{\mathbf{X}} - \mathbf{r})^\top (\Delta\mathbf{w}\hat{\mathbf{X}} - \mathbf{r})\right) + \lambda\Delta\mathbf{w}\mathbf{e}_p^\top + \lambda\mathbf{w}_p \\ &= \text{Tr}(\hat{\mathbf{X}}^\top \Delta\mathbf{w}^\top \Delta\mathbf{w}\hat{\mathbf{X}} - \hat{\mathbf{X}}^\top \Delta\mathbf{w}^\top \mathbf{r} - \mathbf{r}^\top \Delta\mathbf{w}\hat{\mathbf{X}} + \mathbf{r}^\top \mathbf{r}) + \lambda\Delta\mathbf{w}\mathbf{e}_p^\top + \lambda\mathbf{w}_p\end{aligned}\quad (9)$$

To find the local minima of the Lagrangian in Equation 2, we set the partial derivatives with respect to $\Delta\mathbf{w}$ and λ to zero. We first compute the partial derivative with respect to $\Delta\mathbf{w}$:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Delta\mathbf{w}} &= 2\Delta\mathbf{w}\hat{\mathbf{X}}\hat{\mathbf{X}}^\top - 2\mathbf{r}\hat{\mathbf{X}}^\top + \lambda\mathbf{e}_p \\ &= 2\Delta\mathbf{w}\mathbf{H} - 2\mathbf{r}\hat{\mathbf{X}}^\top + \lambda\mathbf{e}_p,\end{aligned}\quad (10)$$

and the partial derivative with respect to λ :

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \Delta\mathbf{w}\mathbf{e}_p^\top + \mathbf{w}_p. \quad (11)$$

Setting the first equation to zero yields:

$$\begin{aligned}\Delta\mathbf{w} &= \mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}^{-1} - \frac{\lambda}{2}\mathbf{e}_p \mathbf{H}^{-1} \\ &= \mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}^{-1} - \frac{\lambda}{2}\mathbf{H}_{p,:}^{-1}.\end{aligned}\quad (12)$$

Setting Equation 11 to zero and substituting $\Delta\mathbf{w}$ from Equation 12, we obtain:

$$\mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}^{-1} \mathbf{e}_p^\top - \frac{\lambda}{2}\mathbf{H}_{p,:}^{-1} \mathbf{e}_p^\top + \mathbf{w}_p = 0. \quad (13)$$

By simplifying with $\mathbf{H}_{p,:}^{-1} \mathbf{e}_p^\top = \mathbf{H}_{pp}^{-1}$, we solve for λ as:

$$\begin{aligned}\lambda &= \frac{2}{\mathbf{H}_{pp}^{-1}}(\mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}^{-1} \mathbf{e}_p^\top + \mathbf{w}_p) \\ &= \frac{2}{\mathbf{H}_{pp}^{-1}}(\mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}_{:,p}^{-1} + \mathbf{w}_p)\end{aligned}\quad (14)$$

Substituting Equation 14 back into Equation 12, we solve for $\Delta\mathbf{w}$ as:

$$\begin{aligned}\Delta\mathbf{w} &= \mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}^{-1} - \frac{1}{\mathbf{H}_{pp}^{-1}}(\mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1} + \mathbf{w}_p \mathbf{H}_{p,:}^{-1}) \\ &= -\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \cdot \mathbf{H}_{p,:}^{-1} + \mathbf{r}\hat{\mathbf{X}}^\top (\mathbf{H}^{-1} - \frac{\mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}}{\mathbf{H}_{pp}^{-1}}).\end{aligned}\quad (15)$$

By recognizing that the expression enclosed in parentheses in the second term corresponds to the Gaussian elimination operation introduced in Section 3, we obtain the final expression for $\Delta\mathbf{w}$:

$$\Delta\mathbf{w} = -\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \cdot \mathbf{H}_{p,:}^{-1} + \mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1}, \quad (16)$$

which is identical to the weight update term in Equation 3.

The next step is to derive the final expression for \mathcal{L} . Substituting Equation 16 into the loss function $\mathcal{L} = \|\Delta\mathbf{w}\hat{\mathbf{X}} - \mathbf{r}\|_F^2$, the loss becomes:

$$\mathcal{L} = \left\| -\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \mathbf{H}_{p,:}^{-1} \hat{\mathbf{X}} + \mathbf{r}\hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} - \mathbf{r} \right\|_F^2. \quad (17)$$

This expression can be expanded as:

$$\begin{aligned}
\mathcal{L} &= \frac{\mathbf{w}_p^2}{(\mathbf{H}_{pp}^{-1})^2} \mathbf{H}_{p,:}^{-1} \mathbf{H} \mathbf{H}_{:,p}^{-1} - 2(\mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} - \mathbf{r}) \left(\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \hat{\mathbf{X}}^\top \mathbf{H}_{:,p}^{-1} \right) + \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \mathbf{H} \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top \\
&\quad - 2\mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top + \mathbf{r} \mathbf{r}^\top \\
&= \frac{\mathbf{w}_p^2}{(\mathbf{H}_{pp}^{-1})^2} \mathbf{H}_{p,:}^{-1} \mathbf{H} \mathbf{H}_{:,p}^{-1} - 2 \frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \mathbf{H} \mathbf{H}_{:,p}^{-1} + 2 \frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{:,p}^{-1} \\
&\quad + \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \mathbf{H} \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top - 2\mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top + \mathbf{r} \mathbf{r}^\top.
\end{aligned} \tag{18}$$

To simplify Equation 18, we observe that the second and third terms are nearly identical, differing only by the presence of the term $\mathbf{H}_{-p}^{-1} \mathbf{H}$. This same pattern appears between the fourth and fifth terms. We begin by simplifying $\mathbf{H}_{-p}^{-1} \mathbf{H}$:

$$\begin{aligned}
\mathbf{H}_{-p}^{-1} \mathbf{H} &= \left(\mathbf{H}^{-1} - \frac{\mathbf{H}_{:,p}^{-1} \mathbf{H}_{p,:}^{-1}}{\mathbf{H}_{pp}^{-1}} \right) \mathbf{H} \\
&= \mathbf{I} - \frac{\mathbf{H}^{-1} \mathbf{e}_p^\top \mathbf{e}_p \mathbf{H}^{-1}}{\mathbf{H}_{pp}^{-1}} \mathbf{H} \\
&= \mathbf{I} - \frac{1}{\mathbf{H}_{pp}^{-1}} \mathbf{H}^{-1} \mathbf{e}_p^\top \mathbf{e}_p.
\end{aligned} \tag{19}$$

With the help of the identity matrix, we can more clearly see how the aforementioned nearly identical terms simplify. Substituting Equation 19 back into Equation 18:

$$\begin{aligned}
\mathcal{L} &= \frac{\mathbf{w}_p^2}{(\mathbf{H}_{pp}^{-1})^2} \mathbf{H}_{p,:}^{-1} \mathbf{H} \mathbf{H}_{:,p}^{-1} + 2 \frac{\mathbf{w}_p}{(\mathbf{H}_{pp}^{-1})^2} \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}^{-1} \mathbf{e}_p^\top \mathbf{e}_p \mathbf{H}_{:,p}^{-1} \\
&\quad - \frac{1}{\mathbf{H}_{pp}^{-1}} \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}^{-1} \mathbf{e}_p^\top \mathbf{e}_p \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top - \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top + \mathbf{r} \mathbf{r}^\top.
\end{aligned} \tag{20}$$

An important observation is that $\mathbf{e}_p \mathbf{H}_{-p}^{-1}$ is in fact an all-zero vector since the p -th row of \mathbf{H}_{-p}^{-1} is eliminated. Consequently, the third term in Equation 20 equals zero. Part of the second term can also be further simplified: $\mathbf{H}^{-1} \mathbf{e}_p^\top \mathbf{e}_p \mathbf{H}_{:,p}^{-1} = \mathbf{H}_{:,p}^{-1} \mathbf{H}_{pp}^{-1}$. The expression $\mathbf{H}_{p,:}^{-1} \mathbf{H} \mathbf{H}_{:,p}^{-1}$ in the first term can also be simplified to \mathbf{H}_{pp}^{-1} . Combining all these simplifications, Equation 20 reduces to:

$$\mathcal{L} = \frac{\mathbf{w}_p^2}{\mathbf{H}_{pp}^{-1}} + 2 \frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{:,p}^{-1} - \mathbf{r} \hat{\mathbf{X}}^\top \mathbf{H}_{-p}^{-1} \hat{\mathbf{X}} \mathbf{r}^\top + \mathbf{r} \mathbf{r}^\top. \tag{21}$$

By reordering the second and fourth terms, we obtain the identical format of \mathcal{L} as presented in Equation 3.

A.2 Proof of Cholesky Decomposition of \mathbf{H}_{-p}^{-1} (Efficient calculation of \mathbf{b} for pruning score \mathbf{S})

Recall that in Section 4.3, we leverage the equality $\mathbf{H}_{-p}^{-1} = \mathbf{L}_{p+1:,p+1} \mathbf{L}_{p+1:,p+1}^\top$ to simplify the computation of term \mathbf{b} for pruning score \mathbf{S} . We provide the proof for this equality using mathematical induction below.

Proof. We begin the mathematical induction proof by establishing the base case: $p = 1$.

Base Case: First, we rewrite the lower-triangular Cholesky factor \mathbf{L} as:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{2:,1} & \mathbf{L}_{2:,2} \end{pmatrix}. \tag{22}$$

The Cholesky decomposition of \mathbf{H} can be written as:

$$\mathbf{H}^{-1} = \begin{pmatrix} \mathbf{H}_{11}^{-1} & \mathbf{H}_{1,2:}^{-1} \\ \mathbf{H}_{2:,1}^{-1} & \mathbf{H}_{2:,2:}^{-1} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{11} & \mathbf{0} \\ \mathbf{L}_{2:,1} & \mathbf{L}_{2:,2:} \end{pmatrix} \begin{pmatrix} \mathbf{L}_{11} & \mathbf{L}_{2:,1}^\top \\ \mathbf{0} & \mathbf{L}_{2:,2:}^\top \end{pmatrix}. \tag{23}$$

Based on this equation, we can construct the following linear system:

$$\begin{cases} \mathbf{H}_{11}^{-1} = \mathbf{L}_{11}^2 \\ \mathbf{H}_{2:,1}^{-1} = \mathbf{L}_{11}\mathbf{L}_{2:,1} \\ \mathbf{H}_{2:,2}^{-1} = \mathbf{L}_{2:,1}\mathbf{L}_{2:,1}^\top + \mathbf{L}_{2:,2}\mathbf{L}_{2:,2}^\top \end{cases}. \quad (24)$$

Solving the above equations, we obtain the expression for $\mathbf{L}_{2:,2}\mathbf{L}_{2:,2}^\top$ as:

$$\begin{aligned} \mathbf{L}_{2:,2}\mathbf{L}_{2:,2}^\top &= \mathbf{H}_{2:,2}^{-1} - \frac{1}{\sqrt{\mathbf{H}_{11}^{-1}}}\mathbf{H}_{2:,1}^{-1}\frac{1}{\sqrt{\mathbf{H}_{11}^{-1}}}(\mathbf{H}_{2:,1}^{-1})^\top \\ &= \mathbf{H}_{2:,2}^{-1} - \frac{1}{\mathbf{H}_{11}^{-1}}\mathbf{H}_{2:,1}^{-1}\mathbf{H}_{1,2}^{-1}. \end{aligned} \quad (25)$$

Recalling the expression for Gaussian elimination introduced in Section 3, Equation 25 essentially represents the removal of the first row and column from \mathbf{H}^{-1} after applying Gaussian elimination to zero out its first row and column. Therefore, we have proven the base case that $\mathbf{H}_{-1}^{-1} = \mathbf{L}_{2:,2}\mathbf{L}_{2:,2}^\top$.

Inductive Step: Assume the statement holds for $p = k - 1$. We show it also holds for $p = k$. Since we know $\mathbf{H}_{-(k-1)}^{-1} = \mathbf{L}_{k:,k}\mathbf{L}_{k:,k}^\top$, the lower-triangular Cholesky factor \mathbf{L} for $\mathbf{H}_{-(k-1)}^{-1}$ can be formulated as:

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_{kk} & \mathbf{0} \\ \mathbf{L}_{(k+1):,k} & \mathbf{L}_{(k+1):,(k+1)} \end{pmatrix}. \quad (26)$$

This formulation is valid because, under our inductive hypothesis for $p = k - 1$, all rows and columns before k have already been removed. We can construct similar linear systems as in the base case:

$$\begin{cases} \mathbf{H}_{kk}^{-1} = \mathbf{L}_{kk}^2 \\ \mathbf{H}_{(k+1):,k}^{-1} = \mathbf{L}_{kk}\mathbf{L}_{(k+1):,k} \\ \mathbf{H}_{(k+1):,(k+1)}^{-1} = \mathbf{L}_{(k+1):,k}\mathbf{L}_{(k+1):,k}^\top + \mathbf{L}_{(k+1):,(k+1)}\mathbf{L}_{(k+1):,(k+1)}^\top \end{cases}. \quad (27)$$

Solving the above equations, we again obtain:

$$\begin{aligned} \mathbf{L}_{(k+1):,(k+1)}\mathbf{L}_{(k+1):,(k+1)}^\top &= \mathbf{H}_{(k+1):,(k+1)}^{-1} - \frac{1}{\sqrt{\mathbf{H}_{kk}^{-1}}}\mathbf{H}_{(k+1):,k}^{-1}\frac{1}{\sqrt{\mathbf{H}_{kk}^{-1}}}(\mathbf{H}_{(k+1):,k}^{-1})^\top \\ &= \mathbf{H}_{(k+1):,(k+1)}^{-1} - \frac{1}{\mathbf{H}_{kk}^{-1}}\mathbf{H}_{(k+1):,k}^{-1}\mathbf{H}_{k,(k+1)}^{-1}. \end{aligned} \quad (28)$$

Again, Equation 28 is equivalent to removing the first row and column of $\mathbf{H}_{-(k-1)}^{-1}$ after applying Gaussian Elimination to zero out its first row and column. This operation is equivalent to computing \mathbf{H}_{-k}^{-1} . Therefore, we have proven that $\mathbf{H}_{-k}^{-1} = \mathbf{L}_{(k+1):,(k+1)}\mathbf{L}_{(k+1):,(k+1)}^\top$. By the principle of mathematical induction, the statement holds for all p with $k \geq 1$.

A.3 Proof of $\mathbf{H}_{:,p}^{-1}/\mathbf{H}_{pp}^{-1} = \mathbf{L}_{:,p}/\mathbf{L}_{pp}$ (Efficient calculation of \mathbf{c} for pruning score \mathbf{S})

This equality has previously been leveraged to simplify the implementation of SparseGPT (Frantar & Alistarh 2023) and GPTQ (Frantar et al. 2022). We also leverage this relationship to simplify the calculation of term \mathbf{c} in our pruning score \mathbf{S} .

Proof. We assume that at the p -th iteration of the calibration, all columns and rows before p have been removed through Gaussian elimination. Based on this assumption, the Cholesky decomposition of the Hessian inverse at the p -th iteration can be formulated as:

$$\mathbf{H}_{-(p-1)}^{-1} = \begin{pmatrix} \mathbf{H}_{pp}^{-1} & \mathbf{H}_{p,(p+1):}^{-1} \\ \mathbf{H}_{(p+1):,p}^{-1} & \mathbf{H}_{(p+1):,(p+1):}^{-1} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_{pp} & \mathbf{0} \\ \mathbf{L}_{(p+1):,p} & \mathbf{L}_{(p+1):,(p+1)} \end{pmatrix} \begin{pmatrix} \mathbf{L}_{pp} & \mathbf{L}_{(p+1):,p}^\top \\ \mathbf{0} & \mathbf{L}_{(p+1):,(p+1)}^\top \end{pmatrix}. \quad (29)$$

This equation also leverages the result proven in Section A.2. Next, we divide the expression $\mathbf{H}_{:,p}^{-1}$ into three parts: $\mathbf{H}_{1:(p-1),p}^{-1}$, \mathbf{H}_{pp}^{-1} , and $\mathbf{H}_{(p+1):,p}^{-1}$. For the first two parts, the equality is straightforward

to prove. Since $\mathbf{H}_{1:(p-1),p}^{-1} = \mathbf{0}^\top = \mathbf{L}_{1:(p-1),p}$, so $\mathbf{H}_{1:(p-1),p}^{-1}/\mathbf{H}_{pp}^{-1} = \mathbf{0}^\top = \mathbf{L}_{1:(p-1),p}/\mathbf{L}_{pp}$. And $\mathbf{H}_{pp}^{-1}/\mathbf{H}_{pp}^{-1} = 1 = \mathbf{L}_{pp}/\mathbf{L}_{pp}$. The only equality that requires proof is: $\mathbf{H}_{(p+1):,p}^{-1}/\mathbf{H}_{pp}^{-1} = \mathbf{L}_{(p+1):,p}/\mathbf{L}_{pp}$. From Equation 29, we can construct the following linear relations:

$$\begin{cases} \mathbf{H}_{pp}^{-1} = \mathbf{L}_{pp}^2 \\ \mathbf{H}_{(p+1):,p}^{-1} = \mathbf{L}_{pp}\mathbf{L}_{(p+1):,p} \end{cases} \quad (30)$$

Substituting these relations back into the expression, it is straightforward to verify that $\mathbf{H}_{(p+1):,p}^{-1}/\mathbf{H}_{pp}^{-1} = \mathbf{L}_{pp}\mathbf{L}_{(p+1):,p}/\mathbf{L}_{pp}^2 = \mathbf{L}_{(p+1):,p}/\mathbf{L}_{pp}$.

A.4 Proof of $\mathbf{U} = ((\Delta\mathbf{X}\hat{\mathbf{X}}^\top\mathbf{L}) \odot \mathbf{M}^u)$

Recall that in Section 4.2, \mathbf{b} is first proposed to be precomputed as $\mathbf{b} = \text{diag}(\mathbf{U}\mathbf{U}^\top)$, where $\mathbf{U}_{p,:} = \Delta\mathbf{X}_{p,:}\hat{\mathbf{X}}^\top\mathbf{L}_{p+1:,p+1:}$. To further improve the algorithmic efficiency, we propose to re-express \mathbf{U} in the form $((\Delta\mathbf{X}\hat{\mathbf{X}}^\top\mathbf{L}) \odot \mathbf{M}^u)$, so that the term $\Delta\mathbf{X}\hat{\mathbf{X}}^\top\mathbf{L}$ can be reused between \mathbf{b} and \mathbf{c} . The proof is straightforward and provided below.

Proof. The p -th row of \mathbf{U} equals $\Delta\mathbf{X}_{p,:}\hat{\mathbf{X}}^\top\mathbf{L}_{p+1:,p+1:}$, where $\mathbf{L} \in \mathcal{R}^{k \times k}$ is the lower-triangle Cholesky factor. For any given vector $\mathbf{z} \in \mathcal{R}^{1 \times k}$ and the Cholesky lower-triangle factor matrix $\mathbf{L}_{p+1:,p+1:}$, we have:

$$(\mathbf{z}\mathbf{L}_{p+1:,p+1:})_i = \begin{cases} 0 & \text{if } i < p+1 \\ \mathbf{z}\mathbf{L}_{:,i} & \text{else } i \geq p+1 \end{cases} \quad (31)$$

Therefore, substituting $\mathbf{z} = \Delta\mathbf{X}_{p,:}\hat{\mathbf{X}}^\top \in \mathcal{R}^{1 \times k}$ into the above equation, we can reformulate the p -th row of \mathbf{U} as:

$$\mathbf{U}_{p,i} = \begin{cases} 0 & \text{if } i < p+1 \\ \Delta\mathbf{X}_{p,:}\hat{\mathbf{X}}^\top\mathbf{L}_{:,i} & \text{else } i \geq p+1 \end{cases} \quad (32)$$

Hence, $\mathbf{U}_{p,:}$ equals $(\Delta\mathbf{X}_{p,:}\hat{\mathbf{X}}^\top\mathbf{L}) \odot \mathbf{M}_{p,:}^u$, where \mathbf{M}^u is a strictly upper-triangular masking matrix with ones above the diagonal. The full computation of \mathbf{U} can thus be re-expressed as $\mathbf{U} = (\Delta\mathbf{X}\hat{\mathbf{X}}^\top\mathbf{L}) \odot \mathbf{M}^u$. By leveraging the independence across rows of $\Delta\mathbf{X}$, $\mathbf{b} = \text{diag}(\mathbf{U}\mathbf{U}^\top)$, where $\mathbf{U} = (\Delta\mathbf{X}\hat{\mathbf{X}}^\top\mathbf{L}) \odot \mathbf{M}^u$.

A.5 Proof of Theorem 1

The key step in proving Equation 8 is to solve for $\mathcal{L}_{\text{SparseGPT}}$. We will follow the same procedure as in Section A.1. According to Equation 1, the optimal weight update for SparseGPT is $\Delta\mathbf{w} = -\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}} \cdot \mathbf{H}_{p,:}^{-1}$. Substituting this into the loss function $\mathcal{L} = \|\Delta\mathbf{w}\hat{\mathbf{X}} - \mathbf{r}\|_F^2$, the loss for SparseGPT becomes:

$$\mathcal{L}_{\text{SparseGPT}} = \left\| -\frac{\mathbf{w}_p}{\mathbf{H}_{pp}^{-1}}\mathbf{H}_{p,:}^{-1}\hat{\mathbf{X}} - \mathbf{r} \right\|_F^2 \quad (33)$$

Given the loss of **DuoGPT** in Equation 17, it is straightforward to observe that the difference between the two loss functions after equation expansion is simply $\Delta\mathcal{L} = \mathbf{r}\hat{\mathbf{X}}^\top\mathbf{H}_{-p}^{-1}\hat{\mathbf{X}}\mathbf{r}^\top$.

The first observation we can make is that, since \mathbf{H}_{-p}^{-1} is positive definite, the loss improvement of **DuoGPT** ($\Delta\mathcal{L}$) will always be greater than or equal to zero. Then, since \mathbf{H}_{-p}^{-1} is obtained from \mathbf{H}^{-1} via Gaussian elimination, it also follows the spectral bound that $\lambda_{\min}(\mathbf{H}_{-p}^{-1}) \geq \frac{\alpha}{\lambda_{\max}(\mathbf{H})}$ where $\alpha > 0$ is the stability constant. We can then rewrite $\Delta\mathcal{L}$ into its quadratic form to get $\Delta\mathcal{L} \geq \frac{\alpha\|\mathbf{r}\hat{\mathbf{X}}^\top\|^2}{\lambda_{\max}(\mathbf{H})}$. Here, it is reasonable to bound $\|\mathbf{r}\hat{\mathbf{X}}^\top\|^2$ as $\|\mathbf{w}\sigma_r\|^2$, where σ_r is just a constant to capture the intensity of the activation residual. This approximation is reasonable due to the fact that we are always perturbing the smallest values during calibration, so the term is not large overall. After assuming a weight norm lower bound of C_w , we can get $\Delta\mathcal{L} \geq \frac{\alpha C_w^2 \sigma_r^2}{\lambda_{\max}(\mathbf{H})}$. Finally, it is intuitive that the activation residual intensity is proportional to the number of "zeroed-out" activations; thus, we include the term $p^x m$ to capture the context of global activation residual intensity. This yields the final error bound as shown in Equation 8.

B Detailed Experimental Setups

This section provides comprehensive implementation details and experimental configurations to ensure reproducibility of our results.

B.1 Calibration Settings for DuoGPT and Unstructured Baselines

We implement all unstructured baselines reported in Table 1 using uniform evaluation settings to ensure fair comparison. Our implementation builds upon the calibration framework from SparseGPT (Frantar & Alistarh 2023), using PyTorch (Paszke 2019).

For both SparseGPT and **DuoGPT**, we enable *act_order*, an option in SparseGPT that sorts weight columns based on Hessian diagonal magnitude to improve pruning performance. The dampening ratio for the Hessian is set to 0.1 for numerical stability. We apply the lazy batch updates and iterative mask blocking techniques from SparseGPT to **DuoGPT** for reducing I/O overhead. The block size and lazy batch size are both set to $B = 128$. For both methods, we prune the input calibration data to each layer based on magnitude before calculating each layer’s Hessians.

We integrate the original pruning and mask selection procedures from Wanda (Sun et al. 2023) into our evaluation framework. Since Wanda lacks a compensation mechanism post-pruning, we maintain dense activation throughout the calibration process (activation sparsity-aware pruning disabled).

For 2:4 structured versions of both SparseGPT and Wanda, we apply mask selection and pruning over every 4-column group with 50% weight sparsity. All other configurations remain identical to their unstructured counterparts. Calibration data remains dense throughout the pruning process for all 2:4 baselines.

B.2 Settings for Speedup Results

We report end-to-end GPU speedup results on an NVIDIA A100 80GB GPU in Table 2 and 6. All speedups are normalized with respect to the dense model baseline rather than reporting absolute running times to ensure fair comparison across different frameworks.

For **DuoGPT**, end-to-end GPU speedup is measured by inheriting TEAL’s custom Triton kernel (Liu et al. 2025) and integrating it into the fast-gpt framework. The corresponding dense model baseline is also evaluated using the fast-gpt framework to obtain normalized speedup ratios. All speedups for structured pruned models are measured using 2SSP’s framework (Sandri et al. 2025), with dense model baselines similarly evaluated within the same framework to compute normalized speedup.

B.3 Settings to Measure the Number of Weights Loaded to SRAM and Model Size

In Figure 3, we report the number of weights loaded into SRAM during inference. For dense models, structured-sparse models, and activation-sparse models, the number of weights loaded into SRAM is fixed. However, for **DuoGPT**, the dual-sparsity approach combined with unstructured weight distribution across rows makes the number of loaded weights a dynamic quantity that depends on which activation patterns are selected. To ensure a conservative and fair comparison, we report the worst-case SRAM loading numbers for **DuoGPT**. Specifically, we sort weight rows by their weight sparsity and assume that dynamic activation sparsity always selects the rows with the lowest weight sparsity (i.e., the densest rows requiring maximum SRAM access). For example, with 40% activation sparsity and 40% unstructured weight sparsity, we first sort all weight rows by their sparsity levels, then select the 40% of rows with the lowest weight sparsity—those requiring the largest number of weights to be loaded into SRAM. This conservative methodology ensures our reported SRAM access numbers represent an upper bound; in practice, **DuoGPT**’s savings on SRAM accesses can be even higher.

Model size results reported in Table 3(b) and Table 5 include the full parameter count, including all model parameters beyond the transformer blocks (e.g., embedding tables and output projection layers).

C Extra Experimental Results

C.1 Extra Performance Comparisons with Structured Pruning Baselines

In Table 6, we provide additional comparison results against structured pruning baselines on LLaMA-2-13B. All structured pruning baselines are pruned with 30% weight-only sparsity. The structured pruning methods use 256 calibration samples (except 2SSP, which uses 32 samples following its original setup) with 2048 token sequences. We employ 50% dual-sparsity for **DuoGPT**. The results demonstrate that **DuoGPT** achieves better performance across speedup, model size, and accuracy metrics. Notably, 50% dual-sparse **DuoGPT** achieves even higher speedup ($1.51\times$) compared to the 30% structured pruned models, where the fastest baseline achieves $1.41\times$ speedup.

Table 6: Extra comparison result with other structured pruning methods on LLaMA-2-13B.

Model	Method	Model Size	Speedup	Wiki2(\downarrow)	PIQA	HellaSw	ARC-E	ARC-C	WinoG	Avg(\uparrow)
LLaMA-2-13B	ShortGPT	9.21B/13.02B	$1.41\times$	39.78	69.80	57.94	52.86	35.75	69.06	57.08
	2SSP	9.21B/13.02B	$1.29\times$	<u>9.07</u>	76.66	70.51	<u>65.28</u>	<u>38.74</u>	<u>68.90</u>	<u>64.02</u>
	BlockPruner	9.21B/13.02B	$1.38\times$	9.67	73.94	64.39	61.87	37.37	66.61	60.84
	DuoGPT	6.67B/13.02B	$1.51\times$	7.17	<u>75.63</u>	<u>69.56</u>	69.95	41.13	68.11	64.88

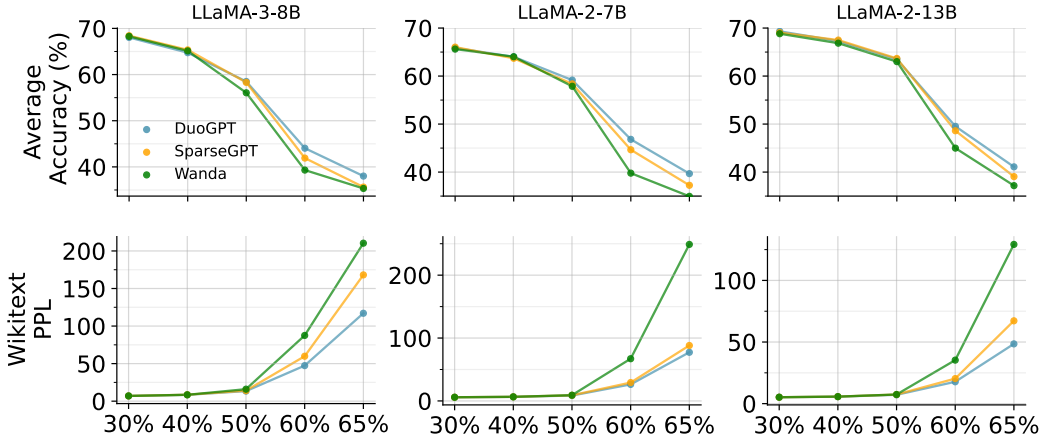


Figure 4: Mean zero-shot accuracy and perplexity on LLaMA-3-8B, LLaMA-2-7B, and LLaMA-2-13B. The results are reported across different dual-sparsity levels. The perplexity is reported for WikiText2 dataset and the accuracy results are averaged across 7 tasks.

C.2 Performance across Different Dual-Sparsity Levels

We provide a comprehensive visualization of our approach’s performance compared to baselines across different dual-sparsity levels in Figure 4. The two baselines are dual-sparse variants of SparseGPT and Wanda, using identical setups as those in Table 1. We plot perplexity (PPL) performance on the WikiText2 dataset and mean zero-shot accuracy across 7 tasks. We visualize sparsity levels of 30%, 40%, 50%, 60%, and 65%. Beyond 65% dual-sparsity, performance degrades too significantly from the dense model to be meaningful, while below 30%, efficiency gains are insufficient to justify the complexity.

Several key trends emerge from Figure 4. At low dual-sparsity regimes ($<50\%$), performance differences between methods are minimal regardless of calibration settings. However, the benefits of activation sparsity-aware calibration become pronounced at high dual-sparsity regimes ($\geq 50\%$). For example, Wanda’s perplexity performance (without activation sparsity-aware calibration) begins to degrade exponentially beyond 60% dual-sparsity. The SparseGPT baseline partially recovers this performance loss through activation sparsity-aware calibration, though these benefits diminish at even higher sparsity levels (65%).

DuoGPT addresses this limitation through output residual compensation, using dense model outputs to compensate for information loss during sparse calibration. This enables superior performance

in extreme dual-sparsity regimes, such as 65%. This trend is consistently observed across different metrics (perplexity and zero-shot accuracy) and model scales, as visualized in Figure 4. Detailed perplexity and zero-shot accuracy results for each task are provided in Table 7 for reference.

Table 7: Detailed zero-shot accuracy results for LLaMA-3-8B, LLaMA-2-7B, and LLaMA-2-13B across different dual-sparsity levels. ARC-E stands for ARC-Easy, ARC-C stands for ARC-Challenge, WinoG stands for WinoGrande, and OBQA stands for OpenBookQA. Bold and underlined values indicate the best and second-best results, respectively.

Model	Method	Wiki2(↓)	PIQA	HellaSwag	ARC-E	ARC-C	WinoG	BoolQ	OBQA	Avg(↑)
LLaMA-3-8B	SparseGPT	7.00	<u>78.94</u>	<u>77.20</u>	<u>76.43</u>	<u>49.06</u>	<u>72.85</u>	<u>81.07</u>	43.40	<u>68.42</u>
	Wanda	6.97	79.27	77.10	75.63	50.17	71.82	80.18	43.80	68.28
	DuoGPT	6.96	78.35	77.08	<u>75.84</u>	48.55	<u>72.14</u>	<u>80.92</u>	<u>43.40</u>	68.04
W _{30%} X _{30%}	SparseGPT	8.57	<u>77.26</u>	<u>73.07</u>	<u>71.68</u>	44.97	<u>70.88</u>	<u>78.81</u>	40.80	<u>65.35</u>
	Wanda	8.46	76.28	72.61	71.42	46.59	<u>69.22</u>	77.58	42.20	65.13
	DuoGPT	8.47	<u>76.88</u>	73.21	70.88	44.28	68.11	78.59	41.20	64.74
W _{50%} X _{50%}	SparseGPT	14.05	72.91	61.96	62.29	37.71	62.43	74.62	36.20	58.30
	Wanda	15.98	71.44	57.01	59.34	35.84	61.33	70.80	36.60	56.05
	DuoGPT	13.41	73.72	<u>61.88</u>	62.96	<u>36.77</u>	64.72	<u>74.25</u>	35.20	58.50
W _{60%} X _{60%}	SparseGPT	<u>59.85</u>	<u>58.32</u>	<u>33.96</u>	<u>39.39</u>	<u>23.38</u>	<u>52.25</u>	<u>58.29</u>	27.80	41.91
	Wanda	87.50	56.64	30.64	35.02	20.82	50.99	55.29	25.80	39.31
	DuoGPT	47.33	61.32	36.95	42.76	24.83	52.57	62.29	<u>27.60</u>	44.05
W _{65%} X _{65%}	SparseGPT	59.85	<u>54.35</u>	<u>28.22</u>	29.76	21.33	<u>50.99</u>	<u>38.56</u>	<u>26.20</u>	35.63
	Wanda	87.50	53.37	28.18	28.79	22.44	50.51	38.04	26.00	35.33
	DuoGPT	47.33	56.75	28.91	32.74	20.56	51.46	49.48	26.20	38.01
LLaMA-2-7B	SparseGPT	5.87	78.45	74.97	73.57	45.39	69.14	<u>77.06</u>	43.60	66.03
	Wanda	<u>5.85</u>	<u>78.29</u>	75.30	<u>73.32</u>	46.16	67.56	76.30	42.40	65.62
	DuoGPT	5.84	77.91	74.66	73.06	<u>45.65</u>	<u>68.43</u>	77.34	44.00	<u>65.86</u>
W _{30%} X _{30%}	SparseGPT	6.52	<u>77.04</u>	71.80	70.12	42.75	66.30	76.12	41.80	63.70
	Wanda	6.50	77.26	72.01	69.99	<u>42.92</u>	67.64	75.60	42.80	64.03
	DuoGPT	6.48	76.61	71.57	71.13	43.34	<u>67.17</u>	76.18	<u>42.00</u>	<u>64.00</u>
W _{40%} X _{40%}	SparseGPT	8.98	74.43	63.89	<u>62.67</u>	35.75	<u>63.46</u>	<u>71.56</u>	36.80	<u>58.37</u>
	Wanda	9.13	73.01	62.30	61.66	35.32	62.75	70.24	39.80	57.87
	DuoGPT	8.58	<u>73.83</u>	<u>63.74</u>	64.18	<u>35.67</u>	65.35	72.54	<u>38.80</u>	59.16
W _{50%} X _{50%}	SparseGPT	29.21	<u>61.15</u>	<u>38.20</u>	<u>42.38</u>	<u>24.57</u>	<u>52.41</u>	63.67	<u>30.20</u>	<u>44.65</u>
	Wanda	67.13	55.93	29.67	34.01	22.70	50.59	58.59	27.00	39.78
	DuoGPT	26.27	63.49	41.54	45.03	26.96	56.59	<u>63.30</u>	30.80	46.82
W _{60%} X _{60%}	SparseGPT	88.17	53.43	29.04	29.25	22.53	49.49	51.38	25.60	37.25
	Wanda	248.9	52.18	28.10	28.16	24.32	50.20	38.26	23.40	34.95
	DuoGPT	77.34	55.44	29.80	32.62	21.59	50.83	58.90	28.60	39.68
W _{65%} X _{65%}	SparseGPT	5.20	79.71	78.62	75.88	49.66	71.90	80.83	46.80	69.06
	Wanda	5.22	79.43	78.56	75.80	49.49	71.35	80.12	47.00	68.82
	DuoGPT	5.18	80.36	78.57	75.46	49.15	<u>71.74</u>	81.77	48.20	69.32
LLaMA-2-13B	SparseGPT	5.70	79.33	<u>76.45</u>	73.36	47.10	71.98	81.10	43.20	67.50
	Wanda	5.69	78.56	76.73	72.43	46.42	68.51	79.88	45.40	66.85
	DuoGPT	5.66	<u>78.67</u>	75.97	<u>73.02</u>	<u>47.01</u>	<u>70.80</u>	<u>80.83</u>	<u>44.80</u>	<u>67.30</u>
W _{30%} X _{30%}	SparseGPT	7.39	<u>76.28</u>	69.14	<u>68.52</u>	41.81	68.67	79.30	42.00	63.67
	Wanda	7.41	77.20	69.15	67.93	40.19	66.54	77.28	42.80	63.01
	DuoGPT	7.17	<u>76.12</u>	69.43	69.49	<u>40.96</u>	<u>67.40</u>	<u>77.83</u>	43.40	<u>63.52</u>
W _{40%} X _{40%}	SparseGPT	20.33	64.74	43.20	49.83	28.33	<u>56.27</u>	68.13	29.80	48.61
	Wanda	35.33	63.00	36.90	44.65	25.09	52.17	62.72	30.40	44.99
	DuoGPT	17.74	66.38	45.74	50.29	28.41	57.06	<u>67.25</u>	31.60	49.53
W _{50%} X _{50%}	SparseGPT	67.21	54.90	29.76	30.77	20.90	49.41	61.87	25.80	39.06
	Wanda	129.22	52.50	29.31	29.04	22.44	51.07	50.70	25.20	37.18
	DuoGPT	48.49	58.16	31.76	34.34	<u>22.35</u>	51.62	62.20	27.20	41.09

C.3 Sensitivity Analysis on the Calibration Set Size and Sequence Length

We present an ablation study to analyze the role of the C4 calibration dataset. We focus on DuoGPT with 50% dual-sparsity using the LLaMA-2-7B model. For the calibration set size study, we fix the sequence length to 2048 tokens. For the calibration sequence length study, we fix the number of calibration samples to 128.

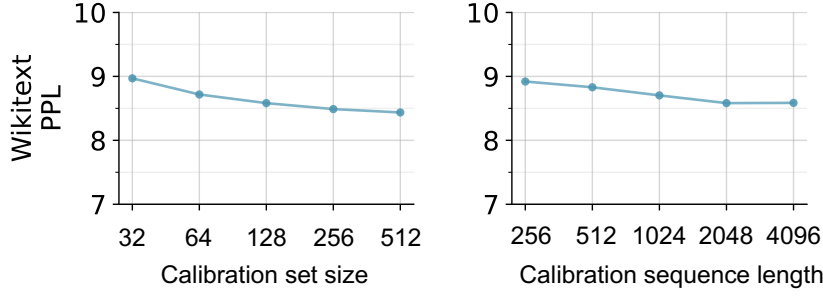


Figure 5: The effect of the C4 calibration set size and sequence length on PPL of WikiText2 dataset for LLaMA-2-7B.

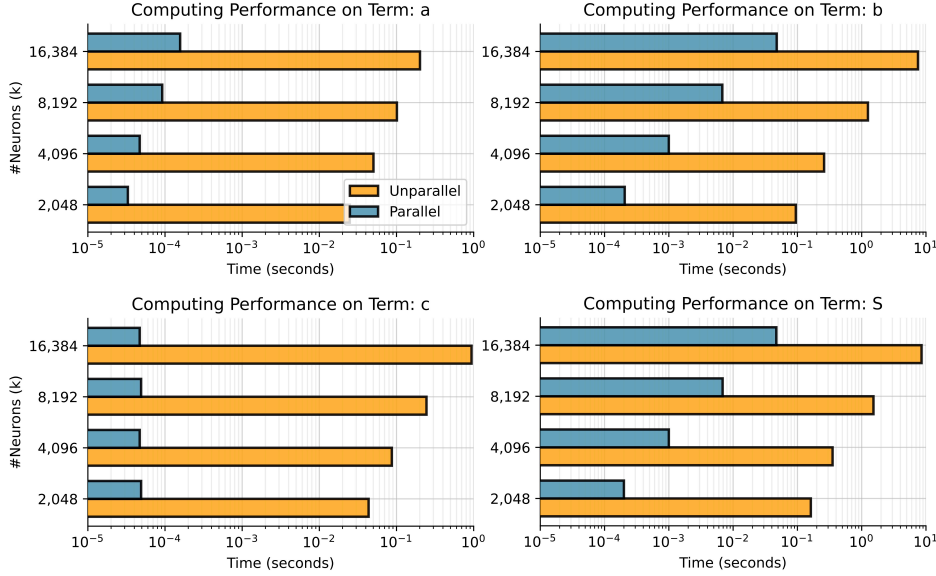


Figure 6: Latency visualization of our algorithm under different size of input channel k .

Figure 5 (left) shows the effect of varying the number of calibration samples on WikiText2 perplexity. The results demonstrate that at least 128 calibration samples provide reasonable performance for our calibration procedure.

We next explore the effect of different sequence lengths in the calibration dataset. Interestingly, we find that beyond a sequence length of 2048 tokens, the perplexity does not continue to decrease, as shown in Figure 5 (right). We conclude that for activation sparsity-aware calibration, a sequence length of 2048 tokens is sufficient to achieve good perplexity performance.

C.4 Algorithm Efficiency

Finally, we provide a visualization of the GPU runtime for our efficient **DuoGPT** implementation (Equation 7) compared to the unparallelized implementation (Equation 6). We measure latency on a single A100 80GB GPU using PyTorch 2.4.0. We provide 10 warm up runs. The token dimension (m) is fixed at 2048. Figure 6 compares the latency for computing a, b, c, and the overall pruning score S. Owing to highly optimized CUDA kernels, our vectorized precomputation of all quantities completes in approximately 1ms for a 4096×4096 layer with $m = 2048$ tokens, achieving roughly $350\times$ speedup over the unparallelized implementation. Note that we do not compare against the naive implementation that follows the optimal pruning order, as it is prohibitively slow and would not yield meaningful comparisons.

C.5 Extra Comparison with Prior Works

Given that CATS (Lee et al. 2024a) is an important prior work that validated the use of a thresholding technique to induce activation sparsity in LLMs, it is important to compare **DuoGPT** against it. We set the comparison to be at 40% global sparsity, which means that **DuoGPT** will have 40% dual-sparsity. Since CATS only explores activation sparsity in FFN layers, it requires 89.7% activation sparsity in those layers to achieve an equivalent 40% global sparsity. The comparison between the two works on 5 downstream tasks is reported in Table 8. Since we directly report the results from CATS, for a fair comparison, we do not report the normalized accuracy for **DuoGPT** in this comparison.

Table 8: Comparison with CATS on LLaMA-2-7B at 40% global sparsity.

Method	PIQA	HellaSw	ARC-E	ARC-C	WinoG	Avg(\uparrow)
CATS	66.27	38.48	45.66	28.16	57.38	47.19
DuoGPT	76.66	53.42	74.37	40.78	67.17	62.48

For reference, we also provide an additional comparison with ReLUfication (Mirzadeh et al. 2023). Since ReLU mostly yields around 50% activation sparsity, we compare it with **DuoGPT** at 22% global dual-sparsity. As discussed in (Lee et al. 2024a), ReLU-based methods perform poorly without enough epochs of fine-tuning.

Table 9: Comparison with ReLUfication on LLaMA-2-7B at 22% global sparsity.

Method	PIQA	HellaSw	ARC-E	ARC-C	WinoG	Avg(\uparrow)
ReLUfication	54.08	25.86	27.95	24.06	48.93	36.18
DuoGPT	78.73	56.80	76.47	43.26	67.96	64.64

C.6 Isolated Effect of Asymmetric Calibration on Weight-only Pruning

It is useful to provide the isolated effect of asymmetric calibration (Li et al. 2025) on weight-only pruning, so that we can get a better idea of how activation sparsity-aware asymmetric calibration helps improve performance. We evaluate two scenarios: (1) weight-only pruning (no activation sparsity), and (2) dual-sparsity (with activation sparsity). We report the WikiText2 PPL results at 50% sparsity across different models in Table 10.

Table 10: Ablation study isolating residual correction effects.

Scenario	Method	LLaMA-2-7B	LLaMA-2-13B	LLaMA-3-8B
Weight-only	SparseGPT	7.11	6.15	9.98
	DuoGPT	6.97	6.03	9.73
Dual-sparsity	SparseGPT	8.98	7.39	14.05
	DuoGPT	8.58	7.17	13.41

The results demonstrate the following: 1. Asymmetric correction is effective on its own: DuoGPT outperforms SparseGPT even in the weight-only pruning setting, showing that the second term in Δw (Equation 3) provides a clear benefit. 2. Activation sparsity amplifies the gain of asymmetric calibration. When activation sparsity is introduced, the magnitude of improvement increases by $2.9\times$, $1.8\times$, and $2.6\times$, respectively, across the three models. These results confirm that our closed-form solution adapts effectively: the asymmetric calibration compensates for error from both weight pruning and activation sparsity.

C.7 Extra Results on Different LLM Architectures.

Demonstrating our method’s generalizability across LLM model architectures is crucial. We have thus evaluated **DuoGPT** on multiple model families beyond LLaMA, which shows consistent

improvements across different architectures. We tested **DuoGPT** on Mistral (Albert et al. 2024), Qwen2 (Bai et al. 2023), and OPT (Zhang et al. 2022) of varying model sizes. We report the WikiText2 PPL at 50% dual-sparsity in Table 11.

Table 11: WikiText2 perplexity results across different LLM model families.

Method	Mist-7B	Qwen-2-7B	Qwen-2-1.5B	OPT-125M	OPT-1.3B
Dense	5.25	7.13	9.54	27.65	14.62
SparseGPT	8.40	10.91	19.50	55.08	35.25
DuoGPT	7.91	10.76	18.11	51.88	31.70

The results show that our method consistently outperforms SparseGPT-based dual-sparse baselines, demonstrating **DuoGPT**'s generalizability across different LLM model architectures.